

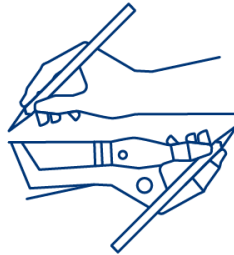


Project: 101094364 — ITHACA — HORIZON-CL2-2022-DEMOCRACY-01

EUROPEAN RESEARCH EXECUTIVE AGENCY (REA)

REA.C – Future Society

C.1 – Inclusive Society



ITHACA
AI To Enhance Civic Participation

ITHACA

artificial Intelligence To enHance Civic pArticipation

D3.4 Final ITHACA platform - prototype

Work Package 3: ITHACA platform design and development

Authors:	KONNEKTABLE
Status:	Final
Due Date:	30/09/2025
Version:	1.0
Submission Date:	20/01/2026
Dissemination Level:	PU - Public

Disclaimer:

This document is issued within the frame and for the purpose of the ITHACA project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101094364. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

This document and its content are the property of the ITHACA Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the ITHACA Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the ITHACA Partners. Each ITHACA Partner may use this document in conformity with the ITHACA Consortium Grant Agreement provisions.

(*) Dissemination level. - Public — fully open (automatically posted online)

Sensitive — limited under the conditions of the Grant Agreement

EU classified —RESTREINT-UE/EU-RESTRICTED, CONFIDENTIEL-UE/EU-CONFIDENTIAL, SECRET-UE/EU-SECRET under Decision 2015/444

ITHACA Project Profile

Grant Agreement No.: 101094364

Acronym:	ITHACA
Title:	artificial Intelligence To enHAnce Civic pArticipation
URL:	https://www.ithaca-project.eu/
Start Date:	01/01/2023
Duration:	36 months

Partners

Short Name	Legal Name	Country
KT	KONNEKT ABLE TECHNOLOGIES LIMITED	IE
CERTH	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	EL
UPAT	PANEPISTIMIO PATRON	EL
RtF	RAISING THE FLOOR	BE
SnP	STAMADIANOS KAI SYNETAIROI DIKIGORIKI ETAIREIA	EL
UniGraz	UNIVERSITAET GRAZ	AT
MNLT	MNLT INNOVATIONS IKE	EL
SIMAVI	SOFTWARE IMAGINATION & VISION SRL	RO
PEDAL	PEDAL CONSULTING SRO	SK
BMA	AGENTIA METROPOLITANA PENTRU DEZVOLTARE DURABILA BRASOV ASOCIATIA	RO
	MESTO MARTIN	SK



Funded by
the European Union

DOCUMENT HISTORY

VERSION	DATE	CHANGES	RESPONSIBLE PARTNER
0.1	04/09/2025	ToC	Epameinondas Koutavelis (KT)
0.2	30/09/2025	Feedback on the structure	Iliana Loi (UPAT), Adrian Dragota (SIMAVI), Katerina Toulidou (CERTH), Michael Bedek (UniGraz), Evangelos Rigas (KT), Epameinondas Koutavelis (KT)
0.3	10/12/2025	Partners Contribution	Iliana Loi (UPAT), Adrian Dragota (SIMAVI), Katerina Toulidou (CERTH), Michael Bedek (UniGraz), Evangelos Rigas (KT), Epameinondas Koutavelis (KT)
0.4	19/12/2025	Review and feedback	Evangelos Rigas (KT), Epameinondas Koutavelis (KT)
0.5	21/12/2025	Internal Review by partners	Katerina Toulidou (CERTH), Adrian Dragota (SIMAVI)
1.0	20/01/2026	Finalisation	Evangelos Rigas (KT), Epameinondas Koutavelis (KT)

Table of Contents

- 1. EXECUTIVE SUMMARY 7
- 2. Introduction..... 9
 - 2.1 Background of the ITHACA Project 9
 - 2.2 Purpose and scope of Deliverable 3.4 9
 - 2.3 Relation to D3.2 and improvements made 10
 - 2.4 Structure of the document 11
- 3. Deployment Overview..... 13
 - 3.1 Hosting environment (cloud, servers, VMs)..... 13
 - 3.2 Deployment architecture (microservices, Docker/Kubernetes)..... 13
 - 3.3 Infrastructure components (databases, brokers, authentication services)..... 14
 - 3.4 Deployment workflows (CI/CD pipelines, monitoring, updates)..... 15
- 4 Final Platform Functionalities 16
 - 4.1 User management (registration, authentication, roles, RBAC)..... 16
 - 4.2 Forum and civic discussion tools..... 18
 - 4.2.1 Topic (Main entity) 18
 - 4.2.2 Proposal 19
 - 4.2.3 Arguments 20
 - 4.3 Gamification dashboard and incentive mechanisms 21
 - 4.4 Argument visualization tools 23
 - 4.4.1 User Experience and Navigation 24
 - 4.4.2 Gamified Participation..... 24
 - 4.5 AI-powered functionalities 24
 - 4.5.1 Multilingual Text Translation 25
 - 4.5.2 Text Summarization..... 26
 - 4.5.3 Language Detection 26
 - 4.5.4 Web crawling tool 26
 - 4.5.5 Toxicity Filtering 27
 - 4.5.6 Sentiment Analysis..... 28
 - 4.6 Personal Information Management System (PIMS) 29
 - 4.6.1 Overview and Role..... 29
 - 4.6.2 Architectural Integration and Workflow 30
 - 4.6.3 Consent Management and User Control Model..... 30
 - 4.6.4 Functionalities and Supported Regulatory Rights 31
 - 4.6.5 Enforcement Across AI and Data Processing Pipelines..... 32

- 5 User Interface Walkthrough 33**
 - 5.1 Login and profile management 33**
 - 5.1.1 Login Page 33
 - 5.1.2 Registration and Profile Initialization 34
 - 5.2 Forum and proposals submission 35**
 - 5.2.1 Forum Access and Content Viewing (Unauthenticated Access)..... 35
 - 5.2.2 Proposal Submission Interface 37
 - 5.2.3 Visualization and Relevance 38
 - 5.3 AI assistant features (summarize, translate, toxicity check) 38**
 - 5.3.1 Summarize 39
 - 5.3.2 Translate 39
 - 5.3.3 Toxicity check..... 40
 - 5.3.4 User Control and Transparency 40
 - 5.4 Gamification features (missions, badges, leaderboard)..... 41**
 - 5.4.1 Technical Implementation of Gamification 44
 - 5.4.2 Technical Implementation of Gamification 44
 - 5.4.3 User Experience 45
 - 5.5 Argument visualization (graphs, pros/cons, votes) 46**
 - 5.6 Data management and transparency features (consent revocation, AI Register) 49**
 - 5.6.1 User Awareness and Transparency 49
 - 5.6.2 Consent and User Control 49
 - 5.6.3 Transparency of AI-Assisted Processing 50
 - 5.6.4 Data Protection, Aggregation, and Analytics 50
 - 5.6.5 Trust, Accountability, and User Confidence 50
 - 5.7 Admin and moderator interfaces..... 50**
 - 5.7.1 Content Analysis and Decision (Approve or Delete)..... 51
 - 5.7.2 Cybersecurity Status 51
 - 5.7.3 AI Fairness 51
 - 5.7.4 Privacy Metrics (PPML) 52
 - 5.7.5 Tools’ Status Overview 52
- 6 Integration and Interoperability 54**
 - 6.1 Backend–frontend integration..... 54**
 - 6.2 API specifications and communication flows 54**
 - 6.3 Data storage and retrieval mechanisms 55**

- 7 Testing and Validation in Pilots 56
 - 7.1 Technical Testing of the Prototype..... 56
 - 7.2 Validation of Platform Functionalities 57
 - 7.3 Feedback Collected During Pilots 57
 - 7.4 Refinements and Adjustments Applied..... 57
 - 7.5 Summary of Testing and Validation Outcomes 58
- 8 Known Limitations and Future Enhancements..... 59
 - 8.1 Current limitations..... 59
 - 8.2 Planned improvements beyond the project lifetime 60
- 9 Conclusions 61
- 10 Annexes..... 62
 - 10.1.1 Deployment configuration files (Docker Compose, Kubernetes manifests) 62
 - 10.1.2 API documentation 75
 - 10.1.3 Code samples for AI-powered toolkit..... 79
 - 10.1.4 Vocabulary and Ontology 82

1. EXECUTIVE SUMMARY

Deliverable **D3.4 – Final ITHACA Platform Prototype** presents the final implementation of the ITHACA platform developed within Work Package 3, building upon the technical foundations, architectural design, and initial operational prototype documented in Deliverable D3.2. This deliverable marks the transition from an early integrated prototype to a **fully functional, pilot-validated platform**, ready to support real-world civic participation scenarios in the pilot cities.

The primary objective of D3.4 is to document the **final architecture, deployment setup, and complete set of platform functionalities**, as implemented and validated during the project's pilot phases. The deliverable demonstrates how the ITHACA platform operationalises artificial intelligence to enhance civic engagement, transparency, and inclusiveness, while ensuring compliance with security, privacy, and ethical AI principles.

Since D3.2, the platform has evolved significantly in terms of **functional completeness, robustness, and usability**. The initial prototype described in D3.2 focused on validating the core architectural approach, containerised deployment, and early versions of key AI components, including topic extraction, toxicity detection, summarisation, translation, and argument visualisation. In D3.4, these components have been **fully integrated, stabilised, and extended**, resulting in a cohesive end-to-end platform supporting user registration, moderated civic discussions, AI-assisted content interaction, gamification-driven engagement, and transparency mechanisms such as the Public AI Register.

The final platform is deployed using a **microservices-based architecture**, leveraging Docker containerisation and a secure reverse-proxy setup to ensure modularity, scalability, and maintainability. Core infrastructure components include MongoDB and PostgreSQL for data persistence, Apache Kafka for asynchronous messaging, and Keycloak for secure identity and access management. Compared to D3.2, deployment workflows have been refined to support reliable updates, debugging, and monitoring within the pilot environments, ensuring operational stability throughout validation activities.

From a functional perspective, D3.4 documents the **complete user journey** across the platform. This includes secure user authentication and role-based access control, structured civic discussions through topics, proposals, and arguments, and AI-powered assistance features such as summarisation, translation, simplification, and automated toxicity moderation. Particular emphasis is placed on the **human-in-the-loop moderation model**, which combines automated AI checks with moderator oversight to balance safety, freedom of expression, and accountability.

A major advancement since D3.2 is the **full operationalisation of the gamification framework**, designed to foster sustained and meaningful civic participation. The final prototype includes missions, experience points, badges, and leaderboards, underpinned by a sigmoid-based level progression model aligned with intrinsic motivation theory. This framework is now fully supported by backend logic and APIs, transforming the earlier conceptual design into a functioning engagement mechanism integrated across platform components.

D3.4 also reports on the **integration, testing, and validation of the platform within the pilot cities**, incorporating multilingual support, evidence extraction from informal web sources, and feedback from real users. Insights gathered during pilot execution have informed refinements to usability, performance, and interaction flows, strengthening the platform's readiness for broader adoption.

In conclusion, Deliverable D3.4 demonstrates that the ITHACA platform has successfully matured from an initial prototype into a **comprehensive, secure, and user-centred civic participation platform**. It consolidates the technical and functional achievements of WP3 and provides a solid foundation for sustainability, scaling, and future exploitation beyond the project lifetime.

2 Introduction

2.1 Background of the ITHACA Project

The **ITHACA (artificial Intelligence To enHance Civic pArticipation)** project is a Horizon Europe initiative that aims to strengthen democratic participation by leveraging trustworthy and human-centric artificial intelligence technologies. The project addresses key challenges in civic engagement, such as low participation rates, information overload, limited inclusiveness, and lack of transparency in decision-making processes, particularly at the local and municipal level.

ITHACA proposes an integrated digital platform that enables citizens to engage in structured civic discussions, express opinions on public issues, and contribute proposals in a safe, inclusive, and transparent environment. By combining AI-powered tools—such as natural language processing, automated moderation, argument visualisation, and conversational assistance—with strong ethical, legal, and privacy safeguards, the platform seeks to enhance both the quality and the impact of citizen participation.

The project is validated through real-life pilot deployments in two European municipalities, ensuring that the developed solutions are grounded in practical needs and adaptable to diverse civic contexts. Within this framework, **Work Package 3 (WP3)** focuses on the **design, development, integration, and deployment of the ITHACA platform**, serving as the technical backbone that enables the project's broader societal objectives.

2.2 Purpose and scope of Deliverable 3.4

Deliverable **D3.4 – Final ITHACA Platform Prototype** complements **Deliverable D3.3**, which documents the **final technical requirements, system architecture, and overall design rationale** of the ITHACA platform. While D3.3 focuses on *what* the platform is, *why* specific design and architectural choices were made, and *how* these choices respond to evaluation outcomes and project objectives, **D3.4 focuses on the concrete technical realisation of those choices in the form of a fully integrated and operational platform prototype.**

The primary purpose of **D3.4** is to present the **final implemented version of the ITHACA platform**, as deployed and validated in the pilot contexts. This deliverable provides a **hands-on, technical view** of the platform, detailing its deployment setup, implemented functionalities, user interfaces, integration mechanisms, and operational behaviour. It serves as the **technical reference document for the prototype**, supporting replication, further development, and potential exploitation of the platform beyond the project lifetime.

In this context, **D3.4 does not replicate in detail** elements that are already comprehensively covered in **D3.3**, such as:

- the conceptual and architectural design of the platform,
- the rationale behind the selection of AI components and services,
- the ethical, legal, transparency, and governance framework underpinning the use of AI within ITHACA.

Instead, these aspects are **referenced where relevant**, with pointers to **D3.3**, and are reflected in D3.4 only insofar as they are manifested in the **actual implementation**. For example:

- the system architecture described in D3.3 is referenced in D3.4 through the concrete deployment configuration and service interactions;
- AI functionalities discussed in D3.3 (e.g. summarisation, moderation, evidence extraction) are presented in D3.4 through their operational behaviour, interfaces, and integration points;
- ethical, transparency, and governance principles defined in D3.3 are reflected in the implemented features such as the Public AI Register, human-in-the-loop moderation workflows, and user-facing transparency elements.

The scope of **D3.4** therefore includes:

- the final deployment and configuration of the platform prototype;
- detailed descriptions of implemented functionalities and user interaction flows;
- technical integration between platform components and external services;
- validation of the prototype through pilot operation and testing activities.

By focusing on the **implementation and operationalisation** of the ITHACA platform, D3.4 provides the necessary technical depth to complement the design-oriented perspective of D3.3, together forming a complete and coherent documentation set for the final outcome of **Work Package 3**.

2.3 Relation to D3.2 and improvements made

Deliverable D3.4 builds directly upon **Deliverable D3.2 – Technical requirements, architecture design and 1st release – prototype**, which documented the initial operational prototype of the ITHACA platform. While D3.2 focused on validating the architectural approach, core technical

requirements, and early integration of key components, D3.4 reflects the **maturation of the platform into a fully operational and pilot-tested system**.

Key improvements and advancements since D3.2 include:

- **Functional completeness:** All major platform components envisioned in earlier deliverables are now fully implemented, including user management, forum-based civic discussions, argument visualisation, AI-assisted content interaction, gamification, and transparency mechanisms such as the Public AI Register.
- **Enhanced AI integration:** AI-powered tools for summarisation, translation, simplification, and toxicity detection have been stabilised, integrated into user workflows, and aligned with human-in-the-loop moderation practices.
- **Operational gamification engine:** The gamification framework has progressed from frontend and conceptual design to a fully implemented backend system, supporting missions, experience points, badges, leaderboards, and a sigmoid-based level progression model.
- **Refined deployment and security setup:** The deployment architecture has been consolidated, with improved container orchestration, reverse proxy configuration, secure authentication, and controlled access to internal services.
- **Pilot-driven refinements:** Feedback from pilot deployments in Brasov and Martin has informed usability improvements, performance optimisations, and functional adjustments.

As such, D3.4 should be viewed not as a replacement but as a **culmination and extension of D3.2**, capturing the final state of the platform after iterative development, integration, and validation.

2.4 Structure of the document

This deliverable is structured to provide a clear and comprehensive presentation of the **final ITHACA platform prototype**, covering its architecture, deployment, functionalities, validation, and future perspectives.

- **Chapter 1 – Executive Summary**
This chapter provides a concise overview of Deliverable D3.4, outlining its objectives, scope, and key outcomes. It summarises the evolution of the ITHACA platform from the initial prototype to the final, pilot-validated implementation and highlights the main technical and functional achievements of Work Package 3.
- **Chapter 2 – Introduction**
This chapter introduces the ITHACA project and its objectives, describes the purpose and scope of Deliverable D3.4, and explains its relation to previous technical deliverables,

particularly D3.2. It also positions D3.4 as the final technical and functional reference for the platform prototype.

- **Chapter 3 – Deployment Overview**

This chapter describes the final deployment environment of the ITHACA platform, including the hosting infrastructure, microservices-based deployment architecture, core infrastructure components, and deployment workflows supporting platform operation.

- **Chapter 4 – Final Platform Functionalities**

This chapter presents the full set of implemented platform functionalities, including user management, civic discussion tools, AI-powered services, gamification mechanisms, argument visualisation, transparency features, and administrative and moderation interfaces.

- **Chapter 5 – User Interface Walkthrough**

This chapter provides a user-oriented walkthrough of the platform, illustrating key interaction flows from login and profile management to participation in discussions, use of AI assistants, gamification features, and moderation tools, reflecting real usage scenarios from the pilot sites.

- **Chapter 6 – Integration and Interoperability**

This chapter details the integration between frontend and backend components, API specifications and communication flows, and data storage and retrieval mechanisms, highlighting how platform components interact internally and with external AI services.

- **Chapter 7 – Testing and Validation in Pilots**

This chapter summarises the testing and validation activities carried out during the pilot deployments, including technical testing, functional validation, user feedback collection, and refinements applied to the platform prototype.

- **Chapter 8 – Known Limitations and Future Enhancements**

This chapter identifies current limitations of the platform prototype and outlines potential improvements and enhancement directions beyond the project lifetime, supporting sustainability and future scalability considerations.

- **Chapter 9 – Conclusions**

The final chapter concludes the deliverable by summarising the overall maturity, readiness, and achievements of the ITHACA platform prototype within WP3.

- **Chapter 10 – Annexes**

This chapter provides supplementary technical material, including deployment configuration files, API documentation, screenshots of the final platform, and pilot feedback logs.

3. Deployment Overview

3.1 Hosting environment (cloud, servers, VMs)

The core of the ITHACA platform is hosted on a dedicated Virtual Machine (VM) located within the SIMAVI premises. This on-premises approach provides direct control over the physical infrastructure, enhancing data sovereignty and security compliance, which is crucial given the civic nature of the platform.

VM Specifications:

- **Operating System:** Ubuntu 23.10 (GNU/Linux 6.5.0-44-generic x86_64)
- **Resources:** The VM is provisioned with substantial resources to handle concurrent user loads and data processing tasks.
 - **Storage:** 500 GB allocation, ensuring ample space for database growth and logs.
 - **Memory:** Efficient usage (~18%) under normal load, with capacity to scale for peak pilot activities.
 - **Networking:** Configured with internal IP addressing (e.g., 10.222.30.234) for secure internal communication.

External access to the platform is managed via a **Reverse Proxy** using **Nginx**, which handles SSL termination and routes traffic to the appropriate internal services. The Nginx configuration ensures that all external traffic is secured via HTTPS (port 443) using Let's Encrypt certificates, automatically redirecting any HTTP requests (port 80) to the secure channel.

3.2 Deployment architecture (microservices, Docker/Kubernetes)

The platform follows a microservices architecture, where distinct functional components are containerized using **Docker**. This approach ensures modularity, ease of updates, and isolation between services.

Containerized Services: The deployment is orchestrated using Docker Compose, defining a network of interacting containers. These services are categorized by their accessibility:

- **Public-Facing Services:** These are exposed to the internet via the Nginx reverse proxy.
 - **Frontend:** The Angular-based user interface accessible at the root domain (/).
 - **Keycloak:** The identity and access management service, accessible via /api/keycloak/.
 - **Communication Service:** Handles real-time interactions and notifications.
- **Internal-Only Services:** These services reside within a private Docker network (app-network) and are not directly accessible from the outside world, enhancing security. They communicate with public-facing services via internal APIs.

- **Backend API:** The core business logic layer (Spring Boot).
- **Audit Service:** Logs critical actions for accountability and transparency.
- **MongoDB:** The primary database for unstructured data and platform content.
- **Orchestrator Service:** Manages complex workflows and inter-service communication.
- **External Integrations (KT Infrastructure):** Services developed by KT (e.g., AI Summarization, Translation) are hosted separately on KT's infrastructure. The SIMAVI-hosted platform communicates with these components securely via RESTful APIs.

3.3 Infrastructure components (databases, brokers, authentication services)

The infrastructure relies on a set of robust, standard components to manage data, identity, and high-throughput messaging:

- **Databases:**
 - **MongoDB:** Deployed as a Docker container (`mongo:latest`), serving as the primary data store for unstructured user content, proposals, and feedback. It is configured with root authentication and persistent volume storage (`mongo-data`) to ensure data durability across container restarts.
 - **PostgreSQL:** Used specifically as the relational backing database for **Keycloak** (`jdbc:postgresql://.../ithaca`), securely storing user credentials, roles, realms, and active session data.
- **Authentication Service (Keycloak):**
 - Identity management is handled by a dedicated **Keycloak** container (`quay.io/keycloak/keycloak:24.0.1`). It is configured for production use with strict HTTPS requirements (`KC_HOSTNAME_STRICT_HTTPS: "true"`) and sits behind the Nginx edge proxy.
 - Custom themes (`ithaca-dashboard`, `custom-ithaca`) are mounted into the container to provide a branded login experience consistent with the platform's visual identity.
- **Message Broker & Real-Time Communication:**
 - **Apache Kafka** is utilized as the central message broker to handle asynchronous, event-driven communication between backend microservices. This ensures reliable data streaming and decoupling of heavy processing tasks.
 - To enable real-time interactivity, specifically for **user-to-user** messaging, the infrastructure leverages **WebSockets**. The Nginx configuration includes specific directives (`at /api/broker/`) to handle the necessary protocol upgrades (`Upgrade: websocket`). This setup allows the platform to push Kafka events directly to the user interface without page refreshes, facilitating seamless, low-latency dialogue.

3.4 Deployment workflows (CI/CD pipelines, monitoring, updates)

The platform utilises a **container-based deployment strategy** to ensure consistency between development and production environments while facilitating maintenance and debugging. Rather than a fully automated pipeline triggering on every commit, the release process focuses on the reliable containerization of compiled artifacts.

- **Backend and Communication Services:** These services are built upon the **bellsoft/liberica-openjdk-debian:17** base image, ensuring a stable Java 17 runtime. The deployment workflow involves compiling the Spring Boot applications into executable JAR files (`ITHACA-0.0.1-SNAPSHOT.jar` and `ithaca-0.0.1-SNAPSHOT.jar`), which are then copied directly into the container's working directory. To support rapid troubleshooting and maintenance within the pilot infrastructure, these containers are configured with **remote debugging enabled** (via `agentLib:jdwp`). The Backend service exposes debug port **5005**, and the Communication service exposes debug port **5006**, allowing technical teams to diagnose issues in the running environment without service interruption.
- **Frontend Service:** The user interface is deployed using a lightweight **node:16-alpine** base image to minimize resource footprint. The Angular application is pre-built into a distribution folder (`dist`) and layered into the container. It is delivered using the **serve** static content server on port **4200**, ensuring high performance for end-users.
- **Update Mechanism:** Updates are managed by generating new build artifacts (JARs or Dist folders) and rebuilding the specific Docker containers. This ensures that updates are atomic and can be rolled back if necessary by reverting to previous container images.

4 Final Platform Functionalities

4.1 User management (registration, authentication, roles, RBAC)

The ITHACA platform leverages **Keycloak**, an open-source Identity and Access Management (IAM) solution, to handle all aspects of user identity. This integration ensures a secure, standard-compliant (OIDC/OAuth 2.0), and centralized mechanism for managing users across the ecosystem.

Registration and Authentication

The platform decouples authentication from the core application logic by delegating these responsibilities to Keycloak.

- **Frontend Integration:** The Angular frontend integrates with the Keycloak client adapter. When an unauthenticated user attempts to access protected resources, the frontend redirects them to the Keycloak login page. This ensures that user credentials are never exposed directly to the client application.
- **Authentication Flow:** The system utilizes the **OpenID Connect (OIDC)** standard, specifically the **Authorization Code Flow with PKCE** (Proof Key for Code Exchange), which is the industry best practice for securing single-page applications (SPAs). Upon successful credential verification, Keycloak issues an **Access Token (JWT)**, a **Refresh Token**, and an **ID Token**.
- **Backend Communication:** The frontend attaches the Access Token to the Authorization header (Bearer Token) of every HTTP request sent to the backend's communication layer. The backend services validate the signature and expiration of this token against the

- **Default User Role:** To streamline the onboarding process, Keycloak is configured with a **Default Role** assignment. Upon registration, every new user is automatically assigned the user role. This role grants basic permissions, such as:
 - Accessing the public forum and reading content.
 - Posting comments and submitting proposals.
 - Managing their own profile data.
- **Moderator Role:** Advanced administrative capabilities are encapsulated in the moderator role. This role is not assigned automatically but is granted administratively to specific trusted users.
 - **Privileges:** Users with this role gain access to the **Moderator Dashboard**, allowing them to review flagged content, delete toxic comments, and oversee the health of the discussion boards.
 - **Role Escalation:** The promotion from user to moderator is handled via the Keycloak Admin Console, where an administrator manually maps the moderator role to the target user’s account. This change is immediately reflected in the user’s token upon the next refresh or login, dynamically unlocking the additional UI components and API endpoints reserved for moderation.

Users > User details

actionadi_moderator

Details | Attributes | Credentials | **Role mapping** | Groups | Consents | Identity provider links | Sessions

→
 Hide inherited roles

<input type="checkbox"/> Name	Inherited	Description
<input type="checkbox"/> default-roles-ithaca	False	`\${role_default-roles}`
<input type="checkbox"/> user	False	-
<input type="checkbox"/> moderator	False	-

4.2 Forum and civic discussion tools

This section describes the functionality of the Civic Discussion and Forum module, which facilitates community engagement, structured proposal submission, and automated content moderation. The core functionality revolves around Topics, Proposals, and Arguments, utilizing a MongoDB backend.

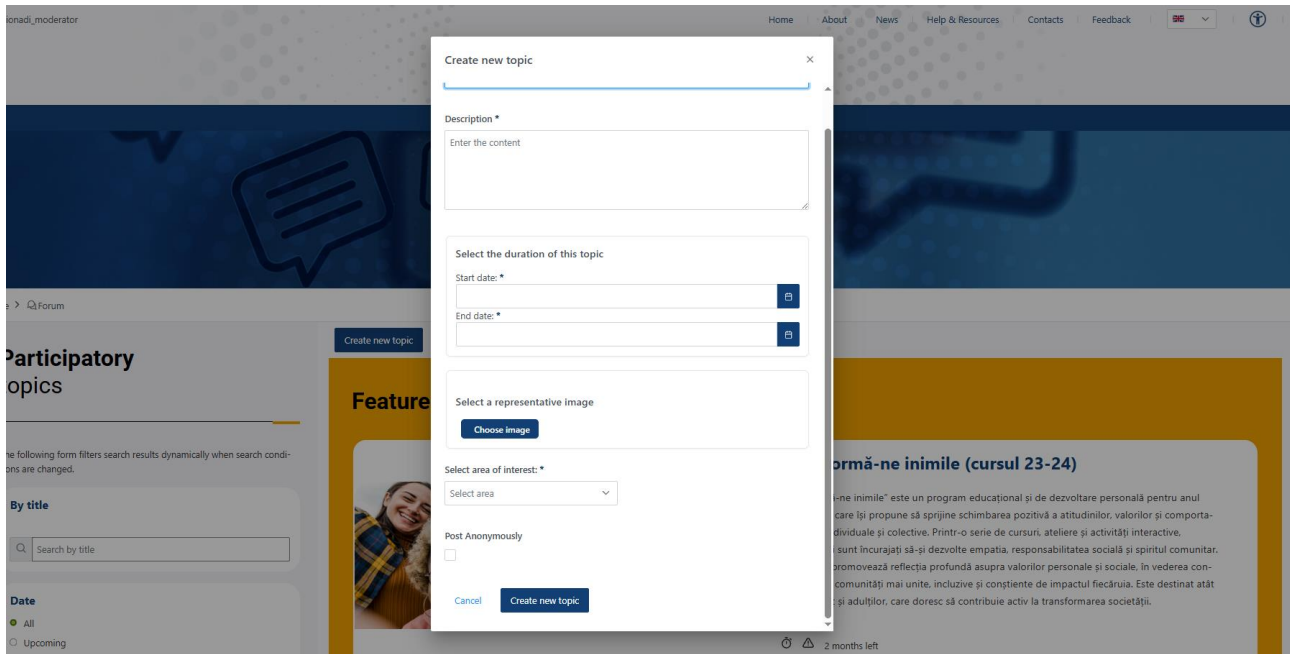
4.2.1 Topic (Main entity)

The Topic entity serves as the central hub for discussion, representing the overall subject or issue.

- **Creation:** New topics can be generated by system **moderators** or automatically **crawled** from predefined external data sources relevant to the municipality.
- **Structure:** Each topic includes metadata fields such as `domainId`, `place` (Brasov/Martin), `type` (topic), `tags` (General, Budget, Transportation, Accessibility) on which topics can be filtered, `name`, `anonymous` (true/false), `title`, and `content`. It has

a summarization field which is updated regularly to reflect a summarization of all the proposal discussions.

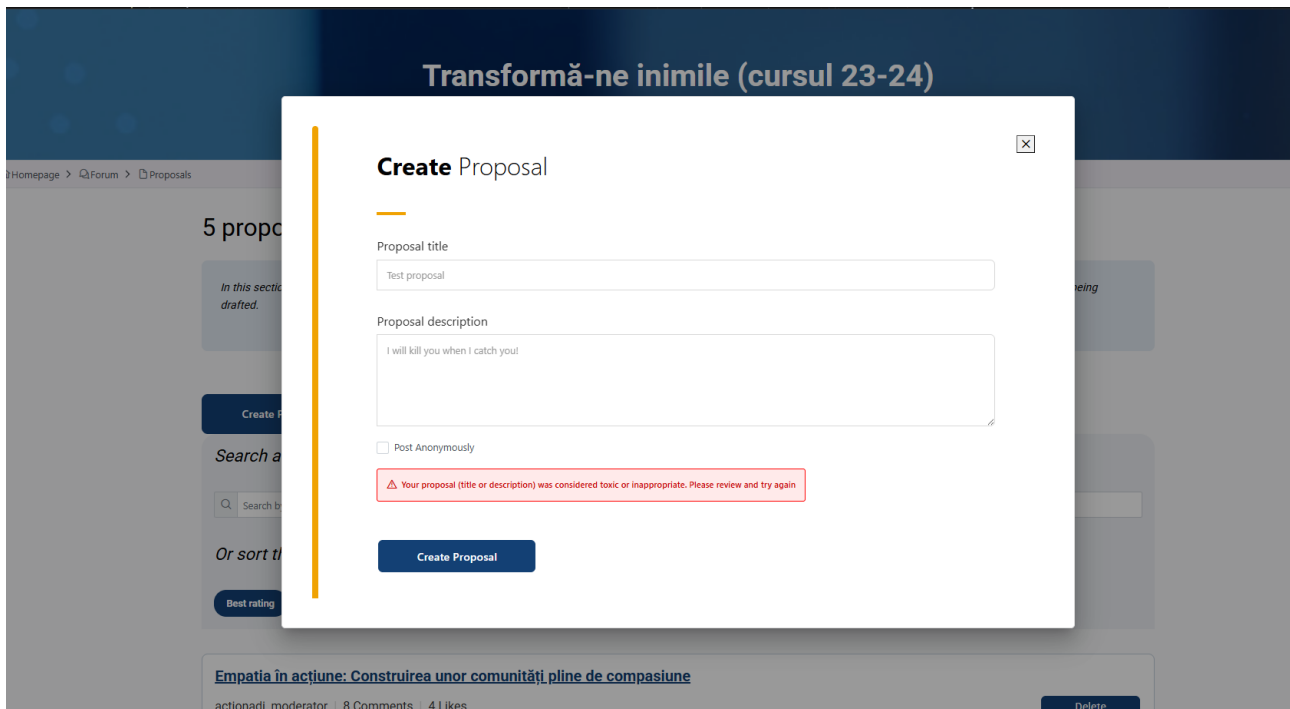
- **Timeline:** the enddate defines the period of the discussion, with the current time remaining calculated dynamically (timeLeft). Can be changed by moderators.
- **Storage:** The topic is stored in the topics MongoDB collection.



4.2.2 Proposal

The Proposal entity represents a suggested solution or idea related to a specific Topic. Proposals are created by **regular users**.

- **Structure:** Proposals are linked to a Topic via `issueId` and include `title`, `description`, `author`, and creation details.
- **Voting Mechanism:** Proposals incorporate a robust voting system:
 - **Tracking:** `userVotes` stores individual user decisions (1 for Upvote, -1 for Downvote).
 - **Tally:** `votes` is a cumulative score reflecting the total balance of votes.
 - **Visualization:** The voting results are visually represented, allowing the community to quickly assess which proposals are the most relevant or favored.
- **Toxicity Evaluation (Content Moderation):**
 - **Pre-submission Check:** Each proposal text is evaluated by an **AI toxicity evaluation tool**.
 - **Rejection:** If a proposal is flagged as toxic, it is rejected, and the user is given the option to modify the text to meet community standards.



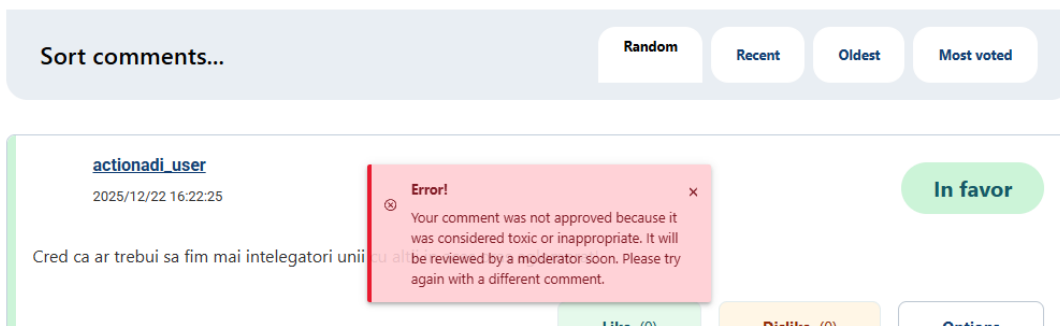
4.2.3 Arguments

The Argument entity allows users to support or critique proposals, creating a detailed chain of discussion.

- **Structure:** Arguments are linked to a specific Proposal via proposalId and include the text, username, and the argument's position.
- **Argument Voting:** Arguments can be individually voted on, helping to surface the most compelling points for and against a proposal.
- **Toxicity and Human-in-the-Loop (HITL) Moderation:**
 - **Initial Check:** Every argument text is checked by the **AI toxicity evaluation tool** before being posted.
 - **Toxic Flagging:** If flagged as toxic, the argument is **not shown** in the regular discussion thread.
 - **Moderator Review:** The flagged argument is rerouted to a human **moderator** for review, who can then manually **approve or reject** the argument.
- **Additional Features:** Arguments also support summarization, translation, and inappropriate content reporting (which triggers moderator evaluation).



2 Comments



4.3 Gamification dashboard and incentive mechanisms

The Gamification System aims to enhance the users’ civic participation and engagement with the ITHACA platform features through gamified mechanisms based on intrinsic motivation theory. The Gamification Dashboard, which is the GUI through which users interact with the gamification system, consists of **three components**:

Gamification Profile

The core of the gamification module lies in the “Gamification Profile”, which can be accessed through the user’s “Profile” section. In the gamification profile, all information related to this module is summarized, including the user’s level, experience points (XP points), and badges earned through completing gamified tasks (missions). The user’s total XP points are computed as the sum of the points awarded from the completed missions, and these points are then used in the calculation of the user’s level as thoroughly described in Chapter 5.4 of this deliverable. In the “Badge List” section, the number of badges per engagement level, namely Competence, Relatedness, and Activity, that the user obtained, is showcased.

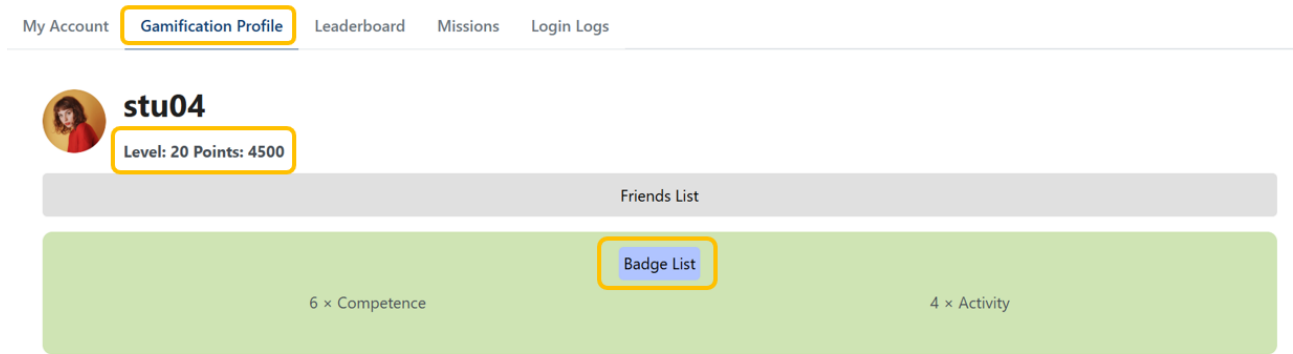


Figure 1: The Gamification Profile

“Missions” Tab

The gamification system’s missions and their respective rewards (XP points and associated badge) as well as their difficulty level and completion status, are outlined in the “Missions” tab (Figure 2). The missions that a user has already completed while engaging with the ITHACA platform are marked with a checkmark and are positioned at the top of the “Missions” list to aid the user in having an overview of already-achieved and available missions.

Name	Description	Reward points	Difficulty	Badge	Completed
Create your 1st post	Write your 1st post	300	Easy	Competence	Yes ✓
Create your 1st comment	Write your 1st comment	300	Easy	Competence	Yes ✓
Evaluate a comment	Evaluate a comment of another user.	300	Easy	Competence	Yes ✓
Democracy in action	Engage with the argument visualization component	400	Medium	Activity	Yes ✓
Vote	Cast a vote on the argument visualization component.	400	Easy	Competence	Yes ✓
Frequent flyer	Login 5 times.	400	Easy	Activity	Yes ✓
Keeping up with the times	Browse through or vote on the 5 most voted proposals.	700	Easy	Activity	Yes ✓
Keeping up with the times (season 2)	Browse through or vote the 10 most recent proposals.	800	Easy	Activity	Yes ✓
Reply on a post	Post a comment on another user's post.	1000	Medium	Competence	Yes ✓
Make a post	Create a post of your own.	1400	Medium	Competence	Yes ✓
Feeling chatty	Engage with the chatbot	400	Medium	Activity	No
Add friends	Expand your civic platform circle by adding 3 people you know.	500	Easy	Relatedness	No

Figure 2: The “Missions” tab. The missions that a user has already completed appear at the top of the “Missions” list and are marked with a checkmark.

Leaderboard

In the “Leaderboard” tab (Figure 3), users can see their ranking compared to other users, based on the XP points collected through mission completion. The top 3 users in terms of XP points are marked with a trophy icon colored based on their ranking (e.g., the 1st user in XP points is marked with a “golden” trophy, the 2nd with a “silver” trophy, etc.).

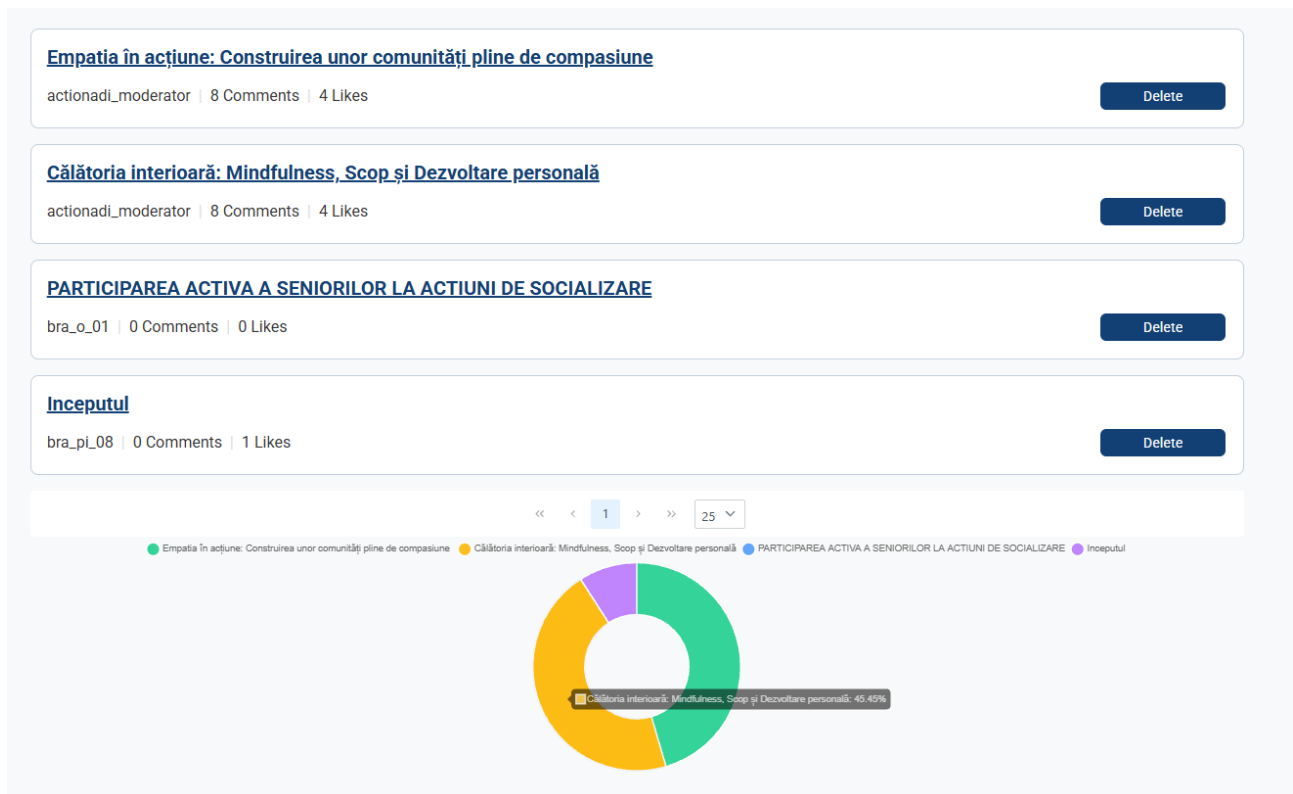
Leaderboard

Username	Points	Reward
stu11	5700	🏆
stu06	5700	🏆
stu02	5300	🏆
stu05	4500	

Figure 3: The Leaderboard

4.4 Argument visualization tools

The Argument Visualization tools are a key feature of the ITHACA platform, designed to help users navigate complex civic debates by structuring contributions into clear **Proposals** and **Arguments** rather than unstructured comment threads. This approach encourages more focused and constructive deliberation.



4.4.1 User Experience and Navigation

- **Accessing the Visualization:** Users access the Argument Visualization tools through the main forum interface. Within a specific topic or issue, users can switch to a "Proposals" view which organizes the discussion around concrete solutions or ideas.
- **Viewing Proposals:** Proposals are presented as individual cards containing a title, description, and author information. Users can browse these proposals to see the different solutions being suggested for the issue at hand.
- **Visualizing Consensus:** A dynamic **Pie Chart** provides a visual summary of the community's sentiment. Each slice of the pie represents a proposal, with the size of the slice corresponding to the net number of votes (upvotes minus downvotes) that proposal has received. This allows users to quickly identify the most popular ideas without having to read through every single post.
- **Interacting with Arguments:** By clicking on a proposal, users can view the detailed arguments associated with it. They can read through the points others have made in support or opposition and can contribute their own arguments to the debate.
- **Voting:** Users can cast votes on both the main proposals and the individual arguments. These votes directly influence the visualization, updating the pie chart and the ranking of items in real-time.

4.4.2 Gamified Participation

To encourage meaningful engagement, the Argument Visualization tools are directly linked to the platform's gamification system.

- **Missions:** Users can complete specific "Missions" by engaging with this tool, such as "Create a Proposal" or "Receive 10 likes on your Proposal."
- **Rewards:** Successfully completing these actions awards the user with XP points and badges, incentivizing them to move beyond passive reading and become active contributors to the civic dialogue.

List of missions						
Name	Description	Reward points	Difficulty	Badge	Completed	
Create your 1st comment	Write your 1st comment	300	Easy	Competence	Yes	✓
Democracy in action	Engage with the argument visualization component	400	Medium	Activity	Yes	✓
Vote	Cast a vote on the argument visualization component.	400	Easy	Competence	Yes	✓
Reply on a post	Post a comment on another user's post.	1000	Medium	Competence	Yes	✓
Positive proposal	Write a city proposal and have it evaluated with 20 positive evaluations.	2300	Hard	Competence	Yes	✓
Create your 1st post	Write your 1st post	300	Easy	Competence	No	
Evaluate a comment	Evaluate a comment of another user.	300	Easy	Competence	No	
Feeling chatty	Engage with the chatbot	400	Medium	Activity	No	
Frequent flyer	Login 5 times.	400	Easy	Activity	No	

4.5 AI-powered functionalities

The **Public AI Register** is a transparency-oriented component of the ITHACA platform, designed to inform users and stakeholders about the artificial intelligence systems employed within the platform.

Its primary objective is to enhance trust, accountability, and understanding by providing accessible information on how AI tools support civic participation.

The register presents high-level descriptions of the AI functionalities used in ITHACA, including their purpose, scope of application, and role within platform workflows. It is intended to support compliance with emerging European AI governance frameworks and aligns with the transparency principles outlined in D3.3.

The Public AI Register includes:

- An overview of AI-enabled functionalities available in the platform.
- Descriptive information aimed at non-technical users.
- A structured layout supporting future extension and maintenance.

The described components are designed to process public and user-generated content in **Romanian, Slovak, and English**, collected from municipal websites and the platform itself. The objective of these AI components is to support safe and respectful public discourse and provide deliberation support.

4.5.1 Multilingual Text Translation

The purpose of this tool is to facilitate the unified processing of data by those AI components that perform optimally in English. Additionally, the translation tool is available inside the platform as standalone text translation for convenience. Supported languages are English, Romanian and Slovak.

The tool uses neural machine translation models based on the **Helsinki-NLP OPUS-MT** family, (publicly available multilingual datasets obtained through Hugging Face) and more specifically:

- *Helsinki-NLP/opus-mt-ROMANCE-en* and *Helsinki-NLP/opus-mt-sk-en*, whenever the target language is English (from Romanian and Slovak respectively)
- *Helsinki-NLP/opus-mt-en-ro* and *Helsinki-NLP/opus-mt-en-sk*, in cases when the language of origin is English (to Romanian and Slovak respectively)

Translation is applied both to crawled content and user-generated content when required. The caveat is that machine translations may be imperfect, especially in areas of political or administrative terminology, which may sometimes lead to misidentifications down the AI pipeline. All models are internally hosted, which means that no third party connection is made to external models, in order to minimize user data processing.

Interested parties can refer to:

- <https://huggingface.co/Helsinki-NLP/opus-mt-ROMANCE-en>
- <https://huggingface.co/Helsinki-NLP/opus-mt-sk-en>

- <https://huggingface.co/Helsinki-NLP/opus-mt-en-ro>
- <https://huggingface.co/Helsinki-NLP/opus-mt-en-sk>
- <https://opus.nlpl.eu/>

for more information.

4.5.2 Text Summarization

Summarization is intended to condense lengthy articles or discussions, and support quick understanding by users of the platform. The intended usage is summarizing the initial text of an article/topic or summarizing the whole topic discussion (initial text and comments).

To achieve this, a summarization pipeline is used, utilizing the model *facebook/bart-large-cnn*. This pre-trained model was specifically chosen because of its multilingual capabilities and the combination of parameters (0.4B params) with relatively lightweight usage. So far, *min_length* and *max_length* token parameters have been set to 30 and 100 respectively, in order to produce concise summaries.

Machine-driven summaries are presented as supportive, not authoritative interpretations, because viewpoints may be omitted during the process. To strengthen against this, original texts remain accessible. Also, the possibility of manipulating min and max length of tokens remains as an extra measure, as the usage of the platform continues. The model is internally hosted, so that third-party connections are avoided to mitigate data processing.

For more information about the model, refer to <https://huggingface.co/facebook/bart-large-cnn>.

4.5.3 Language Detection

Language detection is a complementary tool whose need arises in this multilingual environment. The tool is used internally by other functionalities, such as toxicity text evaluation. The model used for this is *facebook/fasttext-language-identification*, which identifies the Romanian and Slovak languages efficiently.

The tool is self-hosted, in order to support data privacy that might be compromised by third-party connections. More information about the model can be found at: <https://huggingface.co/facebook/fasttext-language-identification>

4.5.4 Web crawling tool

The web crawling component is designed to collect publicly available textual content from municipal websites, such as news articles, announcements, and public notices, serving as primary input for analysis and as a facilitator of collecting municipal information in one place for quick access by users. While not an AI component per se, it feeds into other AI procedures, such as the vocabulary design and topic extraction, therefore it is presented in this section.

The tool is a custom crawler tailored to the HTML structure of participating municipal websites (<https://www.martin.sk/>, <https://www.brasovcity.ro/>). Crawling focuses on article discovery from index pages, extraction of main textual content and preservation of article metadata (such as URL and publication date). These dynamic pages are handled using the headless browser Pypeteer in the background, ensuring that Javascript-rendered content is properly derived.

Crawling avoids excessive request rates, respecting website infrastructure. Potential website structure changes may affect the extraction process, in which case the tool will have to be updated in order to reflect these changes. Only publicly available content is collected.

4.5.5 Toxicity Filtering

In order to better ensure that user-generated content remains respectful and the platform complies with moderation and safety obligations, a toxicity filtering toolkit has been developed. Filtering applies to text and images submitted by platform users.

Text toxicity classification is made available using the pre-trained transformer-based model *unitary/toxic-bert* into a text-classification pipeline. This model was preferred for its multilingual capabilities, in favor of *Hate-speech-CNERG/dehatebert-mono-english*, which as the name implies, works best with english input. Other models that were considered and tested were *google/shieldgemma-2b* and *unitary/multilingual-toxic-xml-roberta*, with slightly less desirable results or hardware-related efficiency than the one finally selected.

When users want to post text, this first passes through the toxicity filter and is evaluated. The result of this evaluation determines whether the text is eventually shown in the platform or not. The tool generates a score of inappropriateness in the range of [0, 1]. After various tests, the threshold has been set to 0.5, which means that anything above that is classified as inappropriate.

Additionally, after input from municipalities, a list of inappropriate words and phrases was compiled. Before jumping into AI-driven evaluation, the toxicity tool cross-checks this list with user-submitted content, and if a match is identified, said content is evaluated as inappropriate (score 1.0). Refining and populating the list even further would help identify harmful text more efficiently.

The tool was tested with safe, unsafe and borderline cases with mostly satisfactory results. Machine-assisted toxicity classification always poses the risks of false positives or false negatives. To mitigate this, and to be up to date with the evolving hate speech or otherwise inappropriate content, two solutions have been proposed:

1. Human moderation, to assist in cases where the tool fails to identify potentially harmful content,
2. Further training of the toxicity filtering model to be carried out, using publicly available toxicity training datasets, such as:
 - a. Jigsaw Toxic Comment Classification Challenge
 - b. Jigsaw Multilingual Toxic Comment Dataset
 - c. Jigsaw Unintended Bias in Toxicity Classification
 - d. Hatebase / Davidson Hate Speech Dataset (*tdavidson/hate_speech_offensive*)

Like with the text toxicity classification tool, the image toxicity classification tool filters content whenever a user tries to post an image inside a discussion. The image is uploaded, resized with the PIL library and evaluated by the vision-language model *openai/clip-vit-base-patch32*, using predefined labels of adult content, violent content, hateful image, safe image. Scores are assigned to each label, with the threshold for safe images having been set at 0.7 or greater (out of 1.0). If flagged as inappropriate, the image is not allowed on the platform. The model was chosen over other candidates such as *google/vit-large-patch16-384* and *openai/clip-vit-large-patch14*, due to better results.

As is the case with the text toxicity classification tool, image classification performed by AI models can produce false positives or negatives. One thing to note is that image classification cannot fact-check or otherwise filter text that may be contained inside an image, rather than classify the visual content of the image. The moderation tool operates as an automated first-line filter, supporting human moderation workflows. Additional training shall be carried out using toxicity datasets specifically designed for images, in order to be on the proactive side of the evolving discourse space, with candidates being:

1. LAION-NSFW / LAION Safety Annotations
2. Hateful Memes Dataset
3. Violence Detection Dataset (VSD)
4. UCF-Crime

Both the text toxicity and the image toxicity tools are self-hosted, in order to avoid data sharing with third-party providers. Interested parties can find more information about the aforementioned models at:

- <https://huggingface.co/unitary/toxic-bert>
- <https://huggingface.co/unitary/multilingual-toxic-xlm-roberta>
- <https://huggingface.co/google/shieldgemma-2b>
- <https://huggingface.co/Hate-speech-CNERG/dehatebert-mono-english>
- <https://huggingface.co/openai/clip-vit-base-patch32>
- <https://huggingface.co/google/vit-large-patch16-384>
- <https://huggingface.co/openai/clip-vit-large-patch14>

4.5.6 Sentiment Analysis

The sentiment analysis component of the platform is designed as a modular Natural Language Processing (NLP) pipeline capable of processing multilingual textual content originating both from externally collected public information sources and from user-generated content within the platform itself. At a high level, sentiment analysis follows a multi-stage processing workflow consisting of text ingestion, language normalization, sentiment inference and result aggregation.

4.5.6.1 Acquisition and normalization

Textual content enters the sentiment pipeline from two main sources: platform-internal content, such as discussion topics, comments, replies, and externally collected content, such as municipal news articles and public announcements retrieved via the web crawling tool (4.3.4).

Given the multilingual context of the platform (Romanian and Slovak), the architecture adopts a pivot-language strategy, using English as the internal processing language for sentiment inference. This decision is motivated by the higher availability and effectiveness of English-language sentiment models, as similarly executed during the creation of the vocabulary dataset (see section 5.1.3 of D3.3 for more details). Language detection and translation are employed, using the tools developed and described in sections 4.6.1 and 4.6.3 of this document, respectively. As a note, translation is applied only for sentiment inference and is not meant to overwrite the original texts.

4.5.6.2 Sentiment Classification and Aggregation

The sentiment analysis layer relies on transformer-based sequence classification models fine-tuned on large-scale opinion datasets. The models considered and tested for this platform are:

- *cardiffnlp/twitter-roberta-base-sentiment*
- *distilbert-base-uncased-finetuned-sst-2-english* (as the name implies, it works best with english text, while inconsistent with Romanian and Slovak; original text translation is therefore crucial for this model)
- *dumitrescustefan/bert-base-romanian-uncased-v1* (an alternative case explored for native Romanian text)

The sentiment analysis pipeline then produces a probabilistic output which offers evaluation based on three sentiment labels: NEGATIVE, NEUTRAL, POSITIVE.

Instead of just assigning sentiment at a single-text level only, the tool can be configured to aggregate sentiment signals across multiple dimensions: per comment, per discussion or per article. Aggregations can then be stored as structured metadata associated with each resource (e.g. discussion), and furthermore, timestamps can be employed to gauge sentiment drift over time.

Sentiment analysis models are self-hosted inside the platform and do not rely on third party connections, in order to mitigate data privacy concerns. For more information about the models, see:

- <https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment>
- <https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english>
- <https://huggingface.co/dumitrescustefan/bert-base-romanian-uncased-v1>

The Public AI Register is designed as a living component, intended to be expanded and refined as the platform evolves and as regulatory requirements mature.

4.6 Personal Information Management System (PIMS)

4.6.1 Overview and Role

The Personal Information Management System (PIMS) is designed as a dedicated module within the ITHACA platform, responsible for enabling users to exercise control over the processing of their personal data in accordance with data protection regulations. PIMS is embedded into the existing platform architecture and interoperates with the external identity and access management infrastructure, i.e. the Keycloak-based authentication system operated outside the platform's domain.

PIMS does not manage user identities, credentials, or authentication directly. Instead, it relies on cryptographically verified identity claims provided via JSON Web Tokens (JWTs) issued by the external authentication provider. This ensures that: identity management remains the responsibility of the platform's central user management system, while PIMS focuses exclusively on consent management, transparency, and enforcement of user data control actions.

4.6.2 Architectural Integration and Workflow

The PIMS is implemented as a Django application module integrated into the main platform backend. It exposes a set of RESTful endpoints that are protected using JWT-based authentication via Django REST Framework SimpleJWT. These endpoints are selectively applied only to PIMS-related views, ensuring that the broader platform API remains unaffected by this authorization logic.

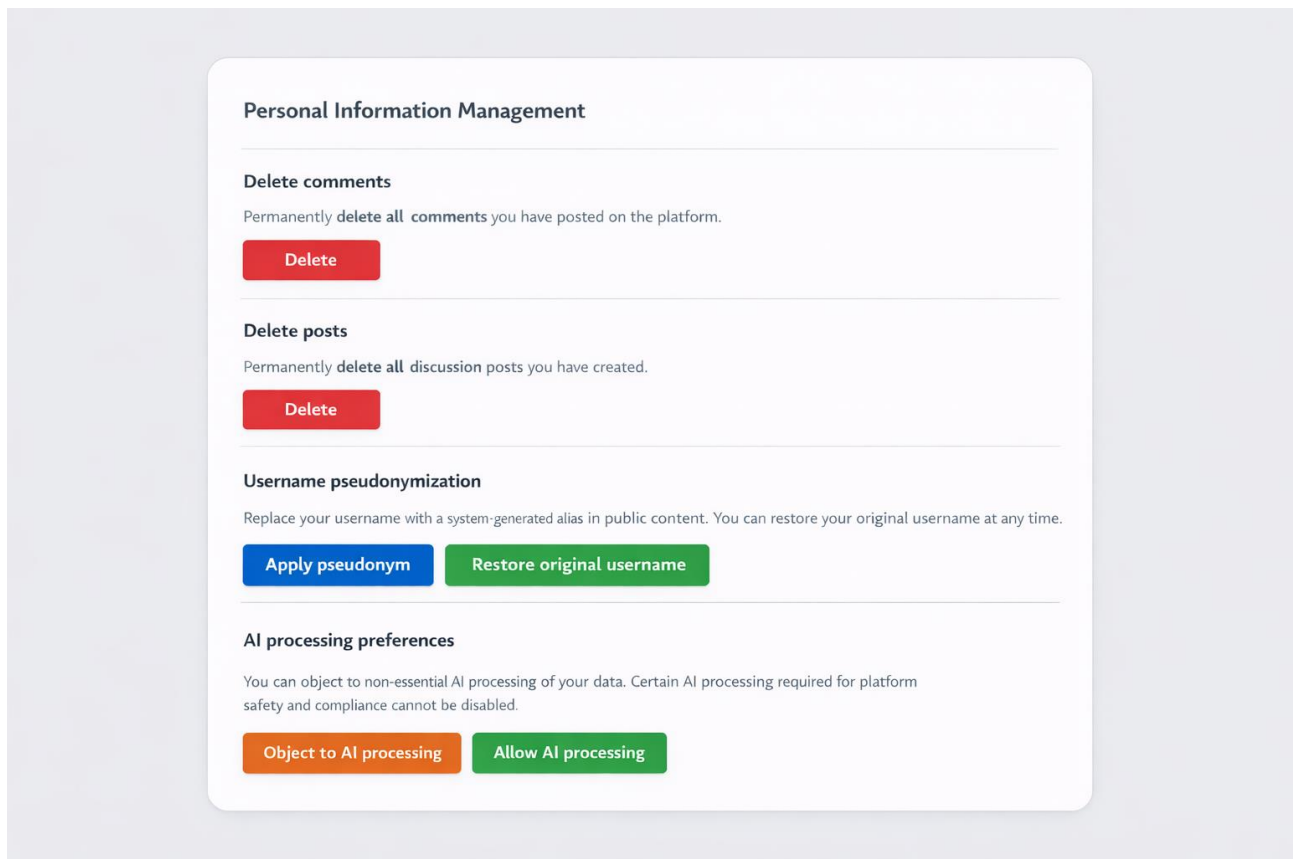
When a user accesses the PIMS interface, requests are authenticated using the same access token issued by Keycloak already in use throughout the platform. The token contains a stable external user identifier, which PIMS treats as the authoritative reference. This identifier is not duplicated or persisted as a local user account. Instead, it is used as a foreign reference when performing consent checks or executing data-related actions.

PIMS actions follow a consistent workflow. First, the user initiates an action through the PIMS interface. The request is then validated at the API level. Finally, PIMS executes the action directly or dispatches it using internal service calls or background tasks, depending on the nature of the operation.

4.6.3 Consent Management and User Control Model

Inside PIMS, consent is structured as an object associated with an externally authenticated user and a specific category of data processing. The system distinguishes between different processing purposes. Users may grant/withdraw consent to those purposes. Each change is timestamped and preserved in an audit log to support accountability and compliance verification.

4.6.4 Functionalities and Supported Regulatory Rights



A. Deletion of User Comments

The “Delete comments” functionality enables users to permanently remove all comments they have posted on the platform. When this action is triggered, PIMS identifies all comment records associated with the user’s external identifier and issues the appropriate deletion requests.

From a regulatory perspective, this functionality primarily supports the right to erasure (Art. 17 GDPR – Right to erasure ('right to be forgotten')). It also contributes to the principle of data minimization under GDPR Article 5.1.c, ensuring that personal data is not retained beyond the user’s expressed intent.

B. Deletion of User Posts

The “Delete posts” action extends the erasure capability to primary discussion resources created by the user. The technical workflow mirrors that of comment deletion, supporting the right to erasure and data minimization.

C. Username Pseudonymization and Restoration

The username pseudonymization feature allows users to replace their publicly visible username with a system-generated alias while preserving internal referential integrity. The original username is never deleted but stored securely and referenced only when the user explicitly chooses to restore it.

This reversible process supports the principle of data protection by design and by default (GDPR Article 25) and contributes to pseudonymization as described in GDPR Article 4.5. While pseudonymization does not equate to anonymization, it significantly reduces the risk of direct identification and supports the broader goal of privacy-enhanced processing.

D. Objection to AI Processing and Consent Reinstatement

The AI processing preference control allows users to object to non-essential AI-based processing of their data (e.g. summarization). When a user objects, PIMS updates the consent object and subsequently the AI pipelines exclude the user's content from processing tasks.

It should be noted though that the system transparently informs users that certain AI-based processing activities, such as toxicity detection and content moderation, cannot be disabled. These activities are classified as necessary for platform safety, legal compliance, and protection of other users.

This functionality supports the right to object (GDPR Article 21) and aligns with emerging transparency obligations under the EU Artificial Intelligence Act, particularly regarding user awareness of automated processing and its purposes.

4.6.5 Enforcement Across AI and Data Processing Pipelines

PIMS is tightly integrated with the platform's AI services through this consent model. Before any AI task is executed on user-generated content, the processing pipeline queries the PIMS consent state associated with the content owner. If consent has been withdrawn or an objection is registered, the task is either skipped or replaced with a compliant alternative (e.g. excluding the data from analysis).

5 User Interface Walkthrough

5.1 Login and profile management

This section describes the key stages of user entry and profile management within the SIMAVI application, powered by Keycloak authentication and customized for the ITHACA project's regional and accessibility requirements.

Login and Profile Management

The SIMAVI platform utilizes **Keycloak** as the primary Identity and Access Management (IAM) provider, ensuring secure login and user profile management. The default Keycloak interface has been customized to meet specific project needs, including regional pilot requirements and accessibility settings.



5.1.1 Login Page

The custom login page allows users to access the platform via several methods:

- **Standard Credentials:** Username (or email) and Password.
- **Social Login:** Users can sign in directly using Google or Facebook accounts.
- **Aesthetics:** The visual display is customized to align with the ITHACA brand identity ("AI To Enhance Civic Participation").

The image shows a registration form for the ITHACA platform. At the top right, there is a language selector showing a UK flag and the text 'English v'. The ITHACA logo, featuring a stylized hand holding a pen and a ruler, is on the left, with the text 'ITHACA AI To Enhance Civic Participation' to its right. The form contains the following fields: 'Username' (text input), 'First name' (text input), 'Last name' (text input), 'Email' (text input), 'Password' (text input), and 'Confirm password' (text input). Below these are two dropdown menus: 'Location' with 'Brasov' selected and 'Accesibility' with 'no' selected. At the bottom left is a blue link '« Back to Login' and at the bottom center is a dark blue button labeled 'Register'.

5.1.2 Registration and Profile Initialization

New users register their accounts through a custom registration page that captures standard user details along with pilot-specific required fields:

- **User Information:** Username, First Name, Last Name, Email, and Password.
- **Pilot Customizations:**
 - **Location:** Users must select their location (e.g., **Brasov** or **Martin**) to tailor content and services according to the local pilot context.

- **Accessibility:** Users can opt-in to accessibility features by selecting **"yes/no"**.
- **Language Selection:** A custom language dropdown allows users to choose between **English, Romanian, and Slovak** for the interface.

5.1.3. Authentication Framework

The underlying authentication is handled by **Keycloak**, which manages the security layer and role assignment (e.g., 'user' or 'moderator'), ensuring that only authenticated and authorized users can access application functionality.

5.2 Forum and proposals submission

This section details the navigation, creation, and visualization features of the platform's core engagement tools: the Community Forum and the Proposal Submission interface. These sections are

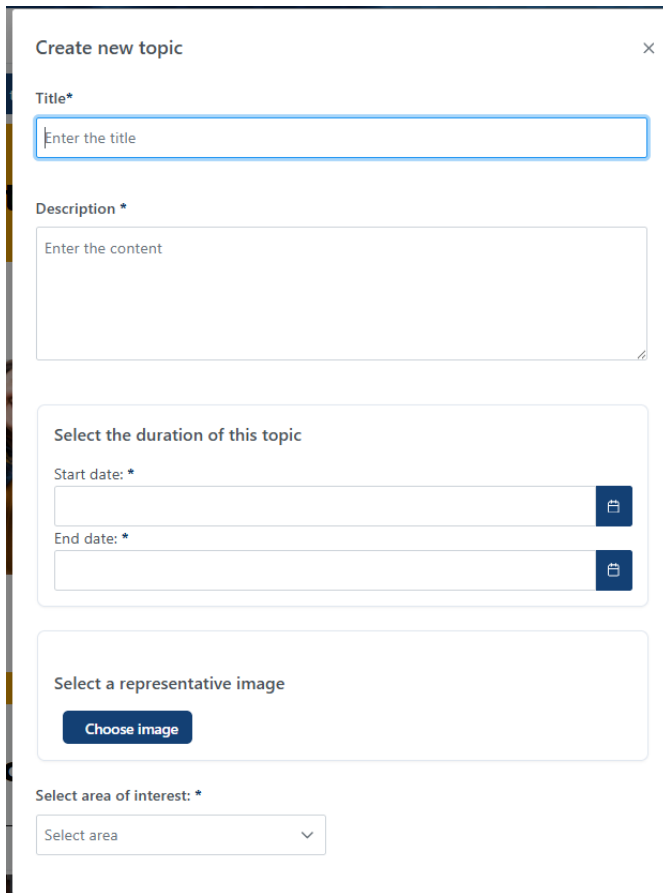
5.2.1 Forum Access and Content Viewing (Unauthenticated Access)

The screenshot displays the forum interface. On the left, a sidebar titled "Participatory topics" contains a search filter section with a "By title" search box and "Date" and "Area" filter sections. The "Date" section has radio buttons for "All", "Upcoming", and "Past". The "Area" section has checkboxes for "All", "General", "Budget", "Transport", and "Accessibility". The main content area features a "Create new topic" button at the top. Below it is a "Featured topics" section with a large orange banner for a topic titled "Transformă-ne inimile (cursul 23-24)". The banner includes a photo of a diverse group of young people and a short description of the program. Below the featured topic is a section for "7 Active topics" with a horizontal scroll of topic thumbnails.

The Forum operates on a tiered access model:

- **Global Access:** The Forum section is prominently accessible from the main header banner, visible on almost any page via a direct link labeled **"community forum"**.
- **Content Visibility:** All users, **including unauthenticated visitors (not logged in)**, can browse and view content:
 - The main Forum page displays and allows filtering of **"Participatory topics"**.
 - Users can view the details of a specific Topic.
 - Users can view the list of existing **Proposals** associated with a Topic.
 - The **pie chart visualization** representing the vote percentages for each proposal is also visible to all users.

Interactive Engagement (Authenticated Access Required)



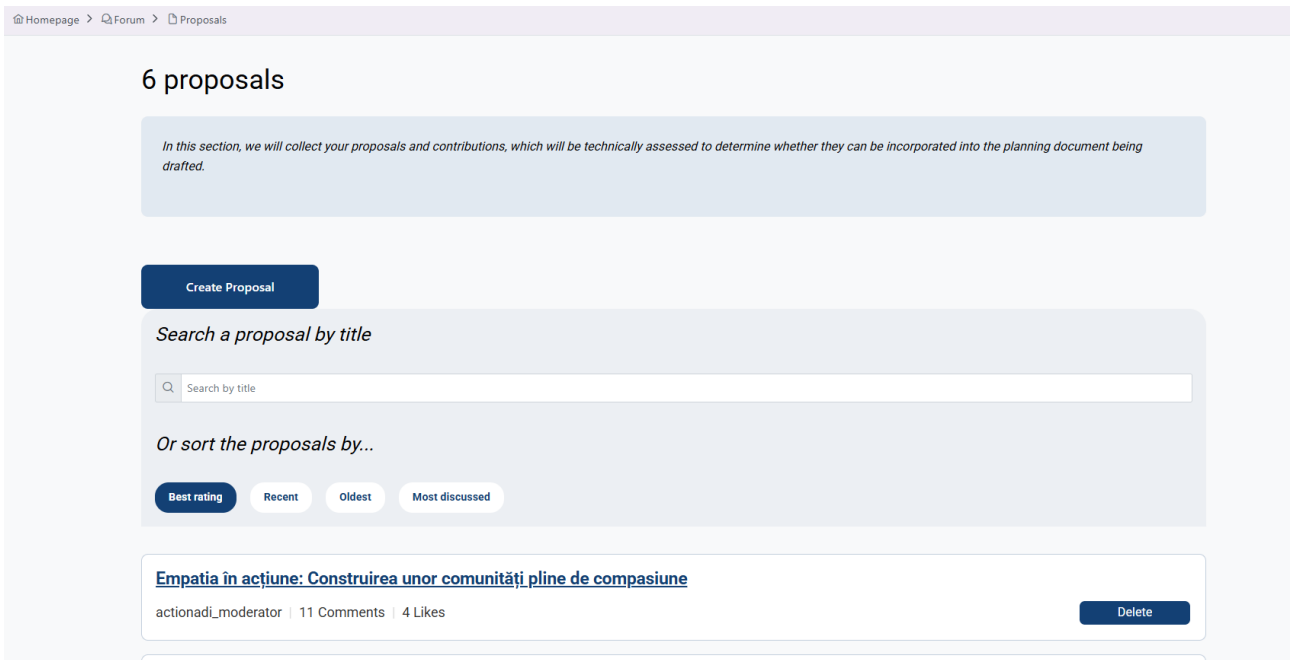
The screenshot shows a 'Create new topic' form with the following fields and controls:

- Title***: A text input field with the placeholder text 'Enter the title'.
- Description ***: A larger text area with the placeholder text 'Enter the content'.
- Select the duration of this topic**: A section containing two date pickers:
 - Start date: ***: A date input field with a calendar icon to its right.
 - End date: ***: A date input field with a calendar icon to its right.
- Select a representative image**: A section with a blue button labeled 'Choose image'.
- Select area of interest: ***: A dropdown menu with the text 'Select area' and a downward arrow.

To ensure accountability and prevent abuse, all interactive functions within the forum require the user to be logged in (authenticated via Keycloak):

- **Topic Creation:** Only moderators (who must be logged in) are permitted to create new discussion topics by clicking the "Create new topic" button.
- **Proposal Submission:** Regular users must be logged in to access the functionality to submit a new proposal via the "Create Proposal" button.
- **Voting:** Interaction features like voting on proposals, liking content, and posting arguments/comments require the user to be logged into their validated profile.

5.2.2 Proposal Submission Interface



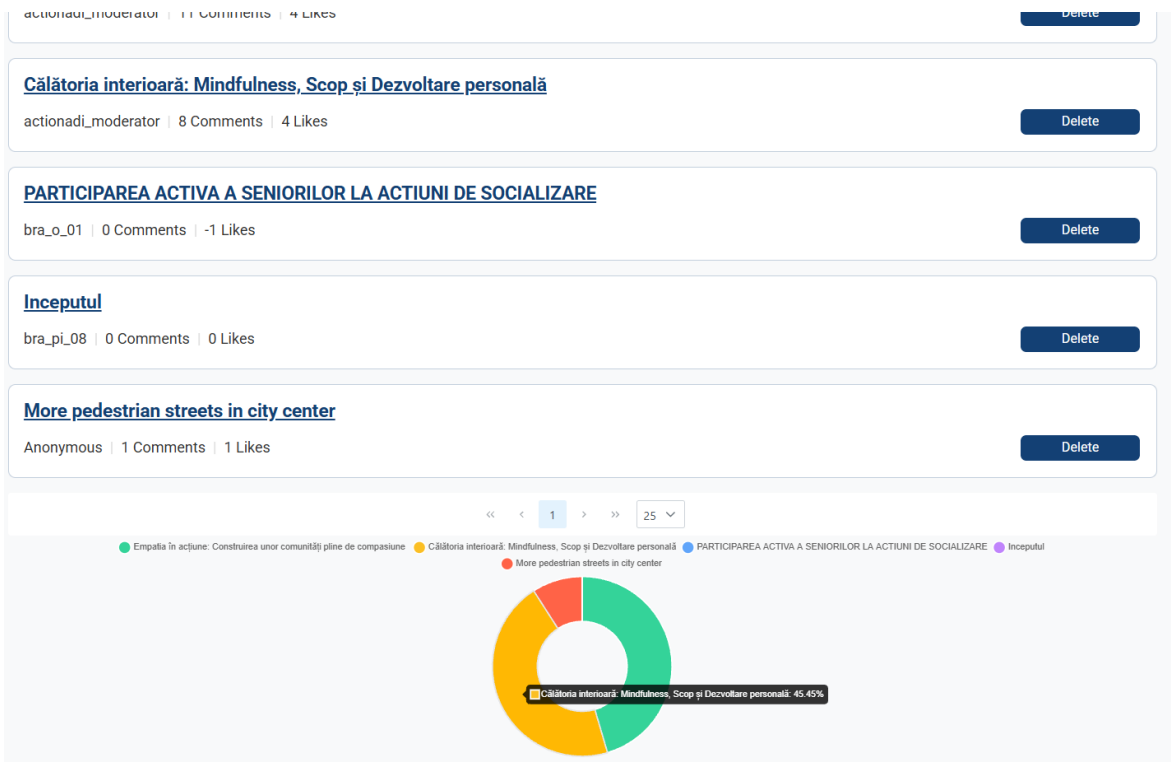
Proposals represent specific solutions or suggestions made by regular users in response to a topic.

- **Proposal List Access:** Once a user clicks on the details of a specific Topic, they can navigate to the Proposals view. This view presents a list of existing contributions, titled "**6 proposals**" in the provided context .
- **Proposal Creation:**
 - To submit a new idea, users click the "**Create Proposal**" button located above the list of proposals.
 - A dialog window appears, prompting the user to enter a short "**Proposal title**" and a detailed "**Proposal description**" .
 - Users also have the option to post their contribution "**Post Anonymously**".
- **Proposal Management:** The proposal list displays key metrics for each entry, including the number of **Comments** and the number of **Likes**.
- Moderators have the right to delete proposals that are deemed not relevant.

The screenshot shows a "Create Proposal" dialog window. It has a title bar with a close button (X). The form contains the following elements:

- A "Proposal title" label followed by a text input field with the placeholder text "Type in a short title of the proposal...".
- A "Proposal description" label followed by a larger text area with the placeholder text "Please type in the description of the proposal...".
- A checkbox labeled "Post Anonymously".
- A dark blue "Create Proposal" button at the bottom.

5.2.3 Visualization and Relevance



Proposal Visualization: At the bottom of the proposal list, a **pie chart visualization** is available .

Voting Percentage: This pie chart visually represents the percentage of votes cast for each proposal relative to the total number of votes.

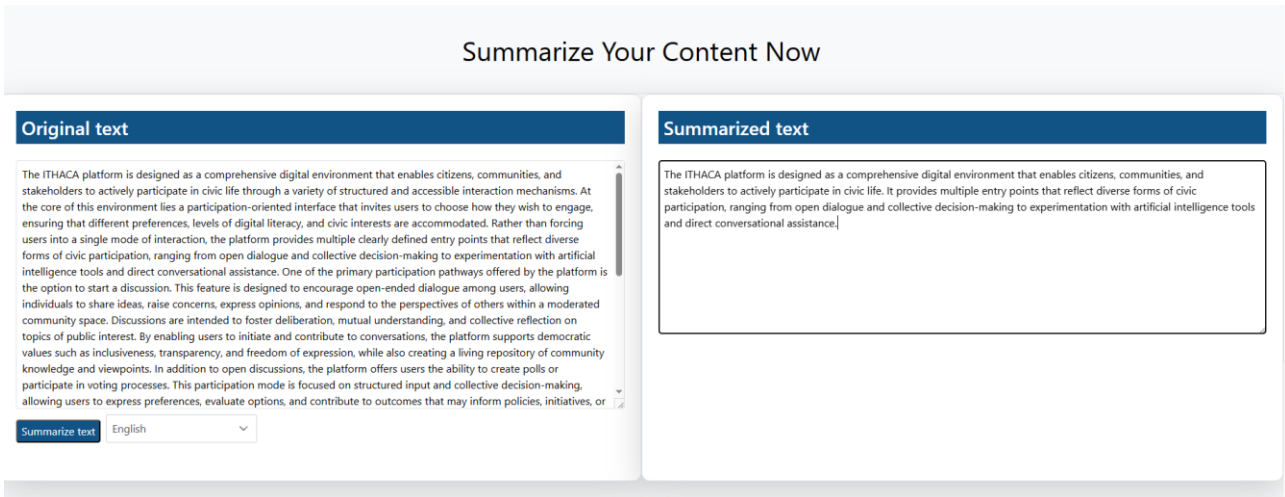
Relevance Indicator: This tool serves to quickly highlight which proposals are the most relevant or favored by the community, such as the proposal "Călătoria interioară: Mindfulness, Scop și Dezvoltare personală," which holds the largest percentage of votes (e.g., 41.67% or 45.45% in the provided examples) .

5.3 AI assistant features (summarize, translate, toxicity check)

The ITHACA platform integrates a set of **AI assistant features** directly into the user interface to support comprehension, accessibility, and ease of participation during civic discussions. These features are designed to assist users while preserving transparency, user control, and access to original content.

The AI assistant functionalities are accessible through contextual action buttons displayed alongside user-generated content, such as topics, proposals, arguments, and comments.

5.3.1 Summarize



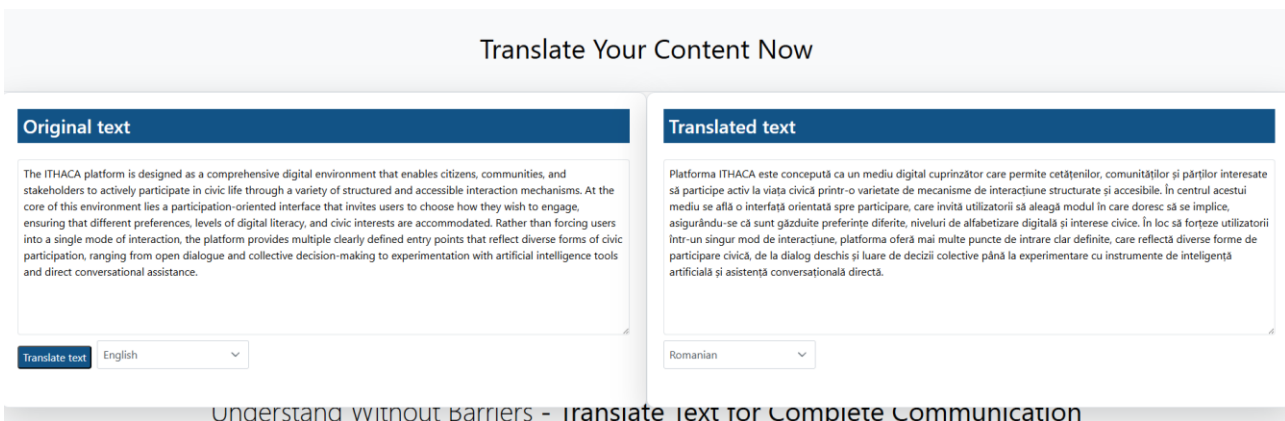
The **Summarise** feature allows users to generate concise overviews of longer content items. By selecting the summarisation option, users can quickly obtain a short textual summary that highlights the main points of a discussion, proposal, or argument thread.

This feature is particularly useful when:

- Topics contain a large number of proposals or comments.
- Proposals include lengthy descriptions.
- Users wish to quickly understand the overall direction of a debate before engaging.

Summaries are clearly indicated as AI-generated and do not replace the original content, which remains fully accessible at all times.

5.3.2 Translate



The **Translate** feature enables users to view content in their preferred language. With a single action, users can translate posts, proposals, arguments, and summaries between the supported platform languages.

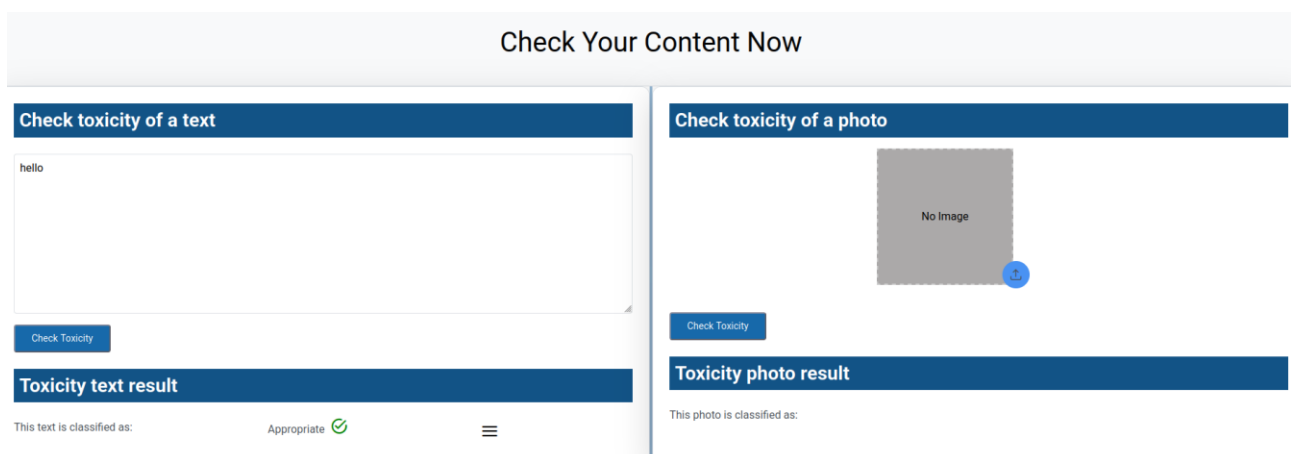
Key characteristics of the translation feature include:

- On-demand activation by the user.
- Preservation of the original text alongside the translated version, when applicable.
- Consistent availability across different content types.

This feature supports multilingual participation and facilitates cross-language interaction among users from different pilot locations.

5.3.3 Toxicity check

The **Toxicity check** feature allows users to check text or images that they would possibly like to submit. Users can type their text in the relevant field or upload an image in the appropriate section and then request evaluation from the toxicity tool (see 4.3.3 for details regarding the tool), with the results being displayed below the submitted content. This way, users can verify that they are submitting safe content and that they contribute to discussions in a respectful manner.



5.3.4 User Control and Transparency

All AI assistant features operate strictly on user request and do not modify content automatically. Each output is clearly labelled as AI-generated, and users are encouraged to consult the original text for full context.

This design ensures that AI acts as a **supportive assistant**, enhancing accessibility and usability without influencing opinions or decision-making processes.

5.4 Gamification features (missions, badges, leaderboard)

The gamification module in ITHACA is designed to foster user engagement and intrinsic motivation through structured rewards and recognition. The system integrates **Missions**, **Badges**, and a **Leaderboard** to incentivize meaningful civic participation.

The tool's main functionality, as described in deliverable D3.1, includes:

- The users being provided with missions, i.e., tasks to fulfill within the platform, which upon completion, they will be rewarded with:
 - Experience points help the user to progress over their level.
 - Associated Badges (similar to trophies).
- Both missions and badges are categorized based on the type of intrinsic motivation they encourage (competence, relatedness, and activity) as well as their difficulty (e.g. bronze badges are awarded to easy missions, etc).
- The users will be ranked among other users based on the total experience points they collect from missions.
- The users' level, total experience points, experience points to progress to the next level, as well as completed missions, obtained and available badges, will be displayed on their profile.

Level Progression Architecture

The platform employs a **sigmoid-based reward function** to model user level progression as a function of total experience (XP) accumulated through mission completion. The use of a sigmoid curve provides a controlled, non-linear growth trajectory in which early levels are attained rapidly to reinforce engagement, while later levels require increasingly larger increments of XP, reflecting the user's deeper mastery and long-term interaction with the system.

To structure the user journey and align motivational dynamics with product objectives, the level system is divided into **three progression phases**:

1. Discovery Phase (Levels 1–4)

During this initial phase, users are introduced to the core functionalities of the platform. The progression curve is intentionally steep, allowing users to gain levels quickly and receive immediate positive reinforcement feedback from the platform, by achieving goals that they themselves have set, by choosing which missions to tackle first. This design supports early engagement, reduces onboarding friction, and encourages exploration of baseline features.

2. Onboarding Phase (Levels 5–14)

In this intermediate phase, users begin interacting with more advanced platform capabilities. The rate of level progression slows moderately to reflect increasing proficiency while maintaining consistent motivational feedback. The system leverages the sigmoid function's mid-growth region to balance user challenge and reward, promoting sustained engagement during the familiarization process.

3. Endgame Phase (Levels 15–50)

The final phase represents deep platform integration, where users have interacted with most functional components. Level advancement becomes significantly more gradual, emphasizing intrinsic motivation rather than rapid progression. The flattening tail of the sigmoid function ensures that users who reach this stage are recognized for sustained participation while preventing level-chasing behavior from overshadowing meaningful platform use.

This phased, sigmoidal progression model supports a structured, psychologically informed user experience that scales with engagement depth while reinforcing long-term interaction with the platform. A visualization of the level progression curve, with respect to the Experience Points the user needs to acquire in order to reach each level is presented in Figure 4.

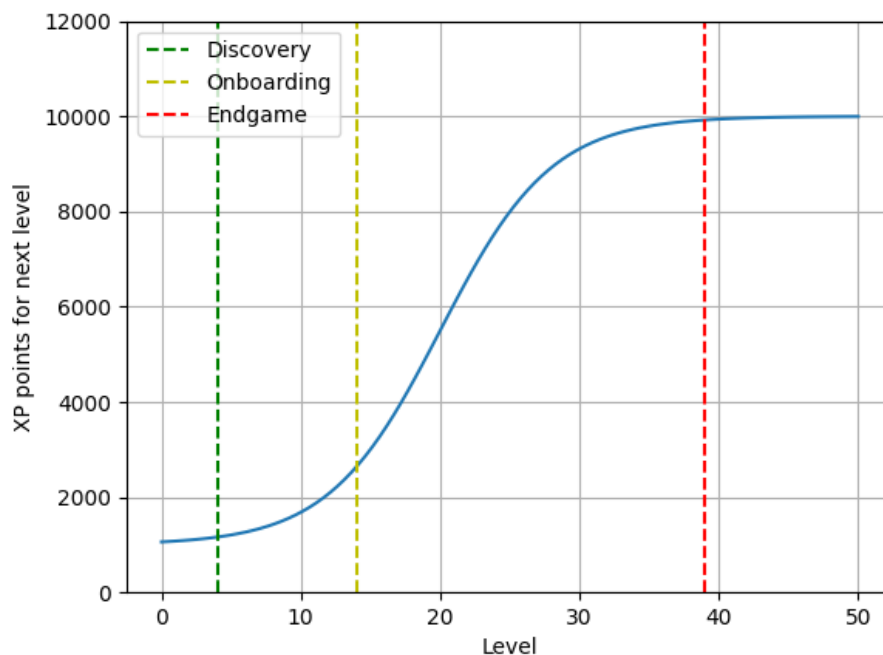


Figure 4

Mission-specific rewards have been implemented through a structured XP allocation framework that assigns experience points based on each mission's difficulty and expected effort. This ensures that the distribution of rewards aligns with the platform's engagement objectives and maintains a balanced progression system. The design enables users to perceive meaningful differences between mission types, reinforcing motivation and supporting a coherent user journey.

The platform also incorporates a sigmoid-based reward function to translate accumulated XP into user levels with a non-linear progression curve. The gamification engine evaluates mission completion criteria, computes total XP earned, and derives the corresponding user level in real time. These functionalities are exposed through a dedicated API, allowing other system components to access XP totals, level information, and mission outcomes seamlessly. This API layer ensures interoperability, scalability, and consistent integration of gamified elements across the broader platform architecture.

The tables presented below summarize the missions integrated into the platform's gamification system. Each mission is designed to encourage user engagement, skill development, and platform

exploration, with rewards and difficulty levels aligned to the expected effort. Missions are categorized by **badge type**: Competence, Relatedness, and Activity, to reflect the specific aspect of user behavior they aim to reinforce.

For each mission, key information is provided, including the mission name, description, reward points, difficulty level, and associated badge. This structured format allows for easy tracking of user progress, reward allocation, and the overall design of the gamified experience.

Table 1: Competence

Name	Description	Reward (XP)	Difficulty
Create your 1st post	Write your 1st post.	300	Easy
Create your 1st comment	Write your 1st comment.	300	Easy
Evaluate a comment	Evaluate a comment of another user.	300	Easy
Vote	Cast a vote on argument visualization.	400	Easy
Reply on a post	Post a comment on another user's post.	1000	Medium
Make a post	Create a post of your own.	1400	Medium
Get positive feedback	Get 10 positive evaluations on your post.	1500	Medium
Comment on a proposal	Comment on a proposal & receive 5 positive evaluations.	1200	Medium
Write a proposal	Create a city proposal of your own.	2000	Hard
Positive proposal	Write a proposal & receive 20 positive evaluations.	2300	Hard

Table 2: Relatedness

Name	Description	Reward (XP)	Difficulty
Explorer's degree	Complete all single acquisition missions.	800	Medium
Add friends	Add 3 people you know.	500	Easy
Go public	Forward a comment or proposal to a contact.	600	Easy
Proposal master	Create 5 proposals & receive 50 total votes.	1500	Hard

Table 3: Activity

Name	Description	Reward (XP)	Difficulty
Democracy in action	Engage with the argument visualization component.	400	Medium
Feeling chatty?	Engage with the chatbot.	400	Medium
Frequent flyer	Login 5 times.	400	Easy
Stay active!	Spend 20 minutes on the platform.	800	Easy
Keeping up with the times	Interact with the 5 most voted proposals.	700	Easy

Keeping up with the times (season 2)	Interact with the 10 most recent proposals.	800	Easy
Maintain a positive post ratio	Make 5 posts with a positive evaluation ratio.	1500	Medium
ENGAGE WITH ALL THE COMPONENTS!!!!	Engage with all components of the platform.	9001	Hard

5.4.1 Technical Implementation of Gamification

All the technical information i.e. the level progression and phases are not available to the user, in order to achieve a more simplified approach, aimed towards people that do not possess significant experience either with social, or with gamification platforms.

The current version of the deliverable extends the initial frontend and planning components by integrating the full backend logic required to operationalize the platform's gamification system. This includes the design and implementation of mission-specific XP rewards, structured according to difficulty tiers to ensure fairness and consistent user motivation. A sigmoid-based progression model has also been incorporated to determine user levels from accumulated XP, enabling a non-linear, psychologically aligned advancement curve. These additions establish the foundational mechanics that govern how users interact with missions, receive feedback, and progress through the platform's experiential phases.

Furthermore, the backend architecture now includes a fully defined gamification engine capable of evaluating mission completion criteria, aggregating XP gains, and computing the user's real-time level. To support seamless communication with the rest of the platform, a dedicated API has been constructed, exposing endpoints for retrieving XP totals, level states, and mission outcome data. This API layer ensures interoperability with existing frontend components and provides a scalable interface for future feature expansion. Collectively, these enhancements transform the system from a conceptual design into a functional, data-driven progression framework integrated across the platform.

5.4.2 Technical Implementation of Gamification

The gamification system is built on a collaborative architecture between the ITHACA platform (managed by SIMAVI) and the Gamification Engine (developed by UPAT).

- **Data Model:** User progress is tracked within the `UserDetails` entity in the ITHACA database. A dedicated `missionData` field stores the status of various user actions relevant to gamification missions. This includes simple counters (e.g., `numberOfLogins`, `numberOfNewFriends`) and boolean flags (e.g., `hasMadePost`, `hasVotedOnArgVis`).
 - **Example Field:** `numberOfLikesOnArgVisProposal` tracks the cumulative likes a user receives on their proposals, directly linking the Argument Visualization tool to the reward system.

- **Mission Evaluation Logic:** The platform employs a real-time evaluation mechanism. Whenever a user performs a gamifiable action (e.g., casting a vote, posting a comment), the backend updates the corresponding field in the user's `missionData`.
 - **External Evaluation Service:** To determine if a mission is completed and to calculate rewards, the platform sends the updated `missionData` payload to the UPAT-developed SIMAVI hosted Gamification Engine via a REST API endpoint (POST `/evaluate_missions`).
 - **Response Handling:** The Gamification Engine processes the user's data against the defined mission criteria and returns a JSON response containing the status of all missions (completed/active), the rewards earned (XP points), and the user's current level.
 - **State Synchronization:** The ITHACA backend then updates the local user profile with the new XP points and mission statuses received from the engine, ensuring the frontend reflects the user's latest achievements immediately.
- **Integration with User Actions:** The gamification logic is embedded directly into the service layer of key features. For instance, the `ProposalService` triggers a mission check (`checkAndCompleteProposalLikesMission`) whenever a proposal is created or voted on. This ensures that the feedback loop between action and reward is seamless and automatic.

5.4.3 User Experience

From the user's perspective, the gamification system provides continuous feedback and clear goals.

- **Missions:** Users are presented with a variety of missions ranging from simple onboarding tasks ("Create your 1st post") to complex engagement goals ("Get 10 positive evaluations from other users"). These missions guide users through the platform's features and encourage high-quality contributions.
- **Badges:** Completing specific categories of missions awards badges (e.g., **Competence, Relatedness, Activity**), visualizing the user's expertise and community standing.
- **Leaderboard:** A dynamic leaderboard showcases top contributors based on their XP, adding a social and competitive dimension that can motivate users to increase their participation.
- **Leveling:** As users accumulate XP from completed missions, their level increases, providing a long-term sense of progression and status within the civic community.

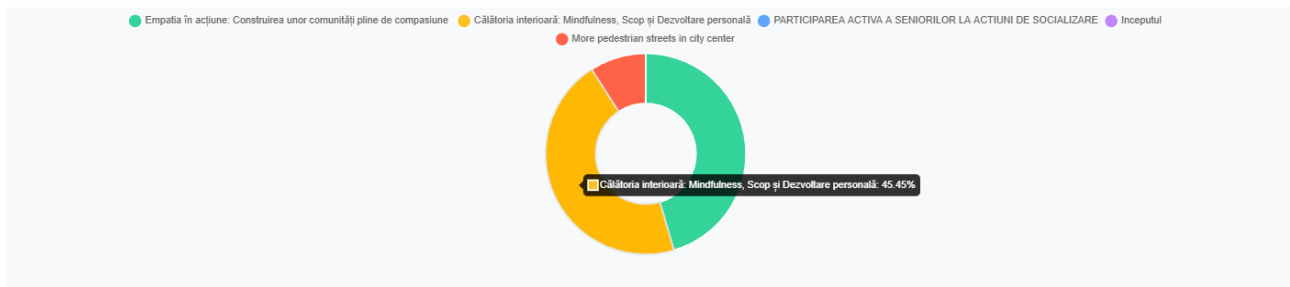
This technical integration ensures that the theoretical gamification models developed by UPAT and UniGraz are translated into a robust, responsive, and engaging user experience on the live platform.

5.5 Argument visualization (graphs, pros/cons, votes)

The Argument Visualization (AV) component is designed to graphically present citizens' suggestions and arguments regarding civic issues initiated by the civic authorities. By visualizing opinions and reasoning, AV helps users understand and engage with different viewpoints, supporting clearer thinking and more effective decision-making. In ITHACA, this component displays a **visual representation of citizens' agreement or disagreement** on various proposals using a **pie chart**, alongside ranked lists of advantages and disadvantages for each idea.

AV is in essence a **forum-like environment** where users/citizens can:

- **Post proposals** regarding a civic issue, suggesting ideas or solutions. Proposals can be submitted by citizens or by users with elevated privileges (e.g., municipal officials testing the platform).
- **Vote for the best proposal**, expressing their support for a particular solution.
- **Post arguments in support of or against a proposal**, indicating agreement or disagreement.
- **Upvote or downvote arguments**, highlighting the most persuasive or relevant contributions.



The AV component uses a **pie chart** to illustrate the distribution of votes across proposals for a given issue. Each segment corresponds to a proposal, with its size representing the level of support. This visualization allows users to quickly identify the most widely accepted ideas. Pie charts are particularly suitable in ITHACA because the number of proposals for a single issue is often small, making heat maps less effective.

Navigating through the AV Component

Participatory topics

The following form filters search results dynamically when search conditions are changed.

By title

Date

All

Upcoming

Past

Area

All

General


Budget

Transport

Accessibility

Create new topic

Featured topics





Transformă-ne inimile (cursul 23-24)

„Transformă-ne inimile” este un program educațional și de dezvoltare personală pentru anul 2023-2024, care își propune să sprijine schimbarea pozitivă a atitudinilor, valorilor și comportamentelor individuale și colective. Printr-o serie de cursuri, ateliere și activități interactive, participanții sunt încurajați să-și dezvolte empatia, responsabilitatea socială și spiritul comunitar. Programul promovează reflecția profundă asupra valorilor personale și sociale, în vederea construirii unei comunități mai unite, incluzive și conștiente de impactul fiecăruia. Este destinat atât tinerilor, cât și adulților, care doresc să contribuie activ la transformarea societății.

🕒 ⚠️ 2 months left

7 Active topics

Users can navigate the AV forum as follows:

1. Select the “Community Forum” Tab

From the ITHACA main menu, click on the “**Community Forum**” tab to access a list of public issues. Each issue includes a short description, allowing users to quickly understand the context (**Figure HERE**).

In this section, we will collect your proposals and contributions, which will be technically assessed to determine whether they can be incorporated into the planning document being drafted.

Create Proposal

Show Summarization

Search a proposal by title

Or sort the proposals by...

Best rating

Recent

Oldest

Most discussed

[Empatia în acțiune: Construirea unor comunități pline de compasiune](#)

actionadi_moderator | 7 Comments | 4 Likes

Delete

[Călătoria interioară: Mindfulness, Scop și Dezvoltare personală](#)

actionadi_moderator | 8 Comments | 4 Likes

Delete

[PARTICIPAREA ACTIVA A SENIORILOR LA ACTIUNI DE SOCIALIZARE](#)

Page 47

2. Viewing Proposals

Selecting an issue opens a screen displaying a list of relevant proposals (**Figure HERE**). For each proposal, users can:

- Read the idea or solution.
- See the total number of votes or level of agreement via the accompanying pie chart.
- Access arguments posted by other citizens in support of or against the proposal.



3. Interacting with Proposals

Upon selecting a proposal, users can:

- **Vote for the proposal** by clicking the like button or equivalent interface element.
- **Post an argument** to indicate agreement or disagreement. Arguments can be upvoted or downvoted by other users to highlight their relevance or persuasiveness.
- **Explore the pie chart**, which dynamically updates to reflect the votes cast by the community.

4. Understanding Advantages and Disadvantages

For each proposal, AV also provides a **ranked list of advantages and disadvantages**, allowing users to evaluate the reasoning behind each idea systematically. This structured display helps citizens weigh pros and cons and understand the collective reasoning behind popular or contested proposals.

Benefits of Using the AV Forum

The Argument Visualization forum empowers citizens and users to:

- Quickly identify widely supported solutions for public issues.

- Participate in informed discussions by reviewing arguments and voting on proposals.
- Understand the reasoning behind different viewpoints through clearly visualized pros and cons.
- Engage collaboratively in civic decision-making, enhancing transparency and inclusiveness.

5.6 Data management and transparency features (consent revocation, AI Register)

The ITHACA platform incorporates a set of **data management and transparency features** aimed at ensuring that users are informed, empowered, and in control of how their data and AI-assisted outputs are handled. These features operationalise the principles defined in the project's Data Management Plan (D7.3) and support compliance with GDPR, ethical requirements, and emerging EU AI governance frameworks.

5.6.1 User Awareness and Transparency

Throughout the user journey, the platform provides clear and accessible information about:

- What types of data are collected (e.g. registration data, user-generated content, interaction logs).
- How data are used to support platform functionality, moderation, analytics, and evaluation.
- Which AI-powered services are involved (e.g. summarisation, translation, toxicity detection).

AI-assisted outputs (such as summaries, translations, or moderation flags) are always clearly marked as **AI-generated**, reinforcing transparency and preventing confusion between human and automated contributions.

5.6.2 Consent and User Control

During registration and profile configuration, users are informed about the purposes of data processing and the applicable legal basis. Where required, **explicit informed consent** is obtained in line with GDPR and the Joint Controllership Arrangement described in D7.3.

From the user interface, individuals can:

- View and manage selected profile and preference information.
- Access information about how their data are used within the platform.
- Exercise control over optional features (e.g. participation settings, accessibility preferences).

These mechanisms are designed to ensure that personal data are processed **only to the extent necessary** for civic participation and platform operation, and always within a secure EU-based infrastructure.

5.6.3 Transparency of AI-Assisted Processing

The platform makes explicit when AI is used to support:

- Content moderation (toxicity detection and flagging),
- Summarisation and simplification of discussions,
- Translation.

In line with the Data Management Plan, AI systems do **not train on ITHACA user data**, and AI outputs are subject to **human oversight**, particularly in moderation-related workflows. This ensures that automated processing remains assistive and accountable, rather than autonomous or opaque.

5.6.4 Data Protection, Aggregation, and Analytics

User-generated content and interaction data are stored securely and retained only for the duration defined in the project's data retention policies. Where data are used for evaluation, reporting, or policy insights, they are processed in **aggregated or pseudonymised form**, ensuring that individual users cannot be re-identified.

The platform does not expose raw personal data to other users or third parties. Any data made available beyond the operational platform (e.g. for research outputs or dissemination) follow the FAIR principles and are anonymised in accordance with D7.3.

5.6.5 Trust, Accountability, and User Confidence

By combining transparent information, user control mechanisms, and clearly communicated AI assistance, the data management and transparency features of the ITHACA platform aim to build **user trust and confidence**. These features ensure that civic participation is supported by digital tools that respect privacy, uphold ethical standards, and remain understandable to non-technical users.

5.7 Admin and moderator interfaces

Moderator Dashboard

The Moderator Dashboard serves as the centralized command center for maintaining a safe and respectful community environment within the ITHACA platform. Access to this page is strictly restricted via Role-Based Access Control (RBAC) to users possessing the **Moderator** role.

The dashboard is divided into two primary functional areas: **Content Management** and the **Statistics Dashboard**.

5.7.1 Content Analysis and Decision (Approve or Delete)

This section facilitates human-in-the-loop moderation for the "Argument Visualization" and "Proposals" components. It functions as a triage center for content flagged through two distinct channels:

- **Automated AI Flagging:** Arguments automatically marked as "toxic" by the integrated Toxicity Detection Tool.
- **User Reporting:** Arguments within a proposal that have been reported by other citizens as inappropriate or offensive.

Workflow: The moderator selects an item from the "List of toxic comments." The interface displays the selected post along with the **Toxicity text result** (the AI classification). The moderator reviews the context and exercises human judgment to take one of the following actions:

- **Delete:** Permanently removes the comment from the platform if it violates community guidelines.
- **Approve:** Clears the flag, making the argument visible to all users in the standard arguments section.
- **Check/Edit:** The interface also allows moderators to sanitize text and re-run the "Check Toxicity" function before approval if minor edits can salvage a constructive point.

5.7.2 Cybersecurity Status

To visualize the outcomes of the AI Cybersecurity tool, a dedicated "**Cybersecurity**" section has been integrated (Feature F in **Figure 6**). This section displays the security status of each AI model integrated into the ITHACA platform.

The tool classifies identified risks and breaches into four severity levels:

- **CRITICAL**
- **HIGH**
- **MEDIUM**
- **LOW**

A "**Help**" button (Feature G) allows the moderator to open a pop-up window (**Figure 7**) containing comprehensive descriptions of these severity levels to aid in decision-making.

5.7.3 AI Fairness

The AI Fairness tool aims to assess fairness conformity in AI systems employed by the ITHACA platform. In the current version of the platform, the tool works in conjunction with the toxicity detection tool to assess whether the latter model exhibits unfair and biased treatment towards vulnerable and marginalized groups, i.e., associates toxic speech with a vulnerable group, with a higher probability. To enhance the impartiality and equal treatment of all users by this system, the AI Fairness tool periodically updates the moderator/user dashboard with model statistics and reports.

D3.4 Final ITHACA platform - prototype

Disparate Impact

Measures whether a specific group receives a less favorable outcome compared to another group. A value of 1.00 indicates perfect fairness.

Equal Opportunity Difference

Evaluates whether the 'true positive' rate (e.g., success) is similar for different groups.

Generalized Entropy Error

A metric that quantifies inequality or uncertainty in a distribution, applied here to measure fairness distribution.

Statistical Parity Difference

Indicates the difference in the positive rate (e.g., selection) between groups.

Treatment Equality

Examines whether error rates (false positives and false negatives) are equal across different groups.

2. **Tools Status:** This section presents the operational, security, and privacy status of essential AI tools integrated into the platform. Each tool, such as **AI Fairness**, **Privacy Preserving Machine**, and **AI Cybersecurity**, is periodically evaluated. You will see the latest update and the status of each aspect:

Security

Reflects the tool's robustness against cyber threats and data integrity. The color (Green, Orange, Red) indicates the level of compliance or alerts.

Privacy

Evaluates the tool's compliance with data protection and privacy standards.

See the below statistics for both sections:

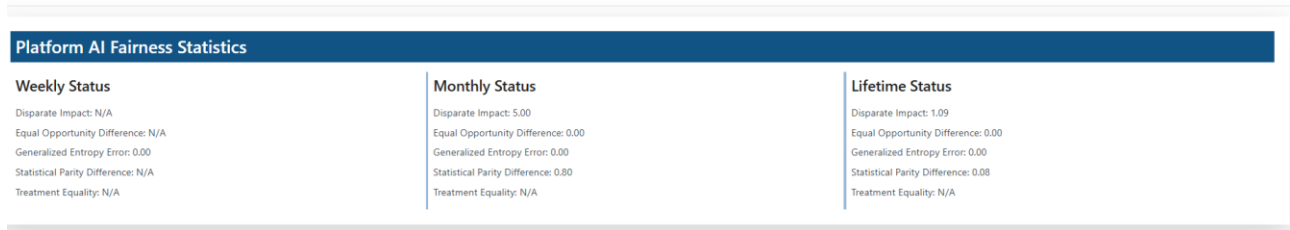


Figure 7

5.7.4 Privacy Metrics (PPML)

The “**Privacy**” section (Feature H in **Figure 8**) visualizes the effectiveness of the differential privacy methods. To maintain impartiality and fairness, the author of any selected post remains anonymous.

This section displays the best values (e.g., obtained after 50 training epochs) for metrics derived from threshold membership inference attacks:

- **Area Under Curve (AUC)**
- **Attacker Advantage**
- **Positive Predictive Value (PPV)**

Moderators can view detailed graphical representations of these metrics by clicking the “**PPML Metrics**” button (Feature I), which opens a pop-up window containing the relevant plots (**Figure 9**). These graphs provide insight into the model's vulnerability to privacy attacks. As these metrics are intended to be updated periodically (e.g., weekly or monthly), a “**Refresh Metrics**” button (Feature J) is provided to update the values displayed.

5.7.5 Tools' Status Overview

The interface includes a “**Tools' Status**” table (Feature K in **Figure 10**) that provides a high-level operational overview of the evaluation tools (AI Fairness, PPML, and AI Cybersecurity). It lists the date of the last update and uses a traffic-light system to indicate status:

- **Green:** Fully Operational / Secure / Private.
- **Orange:** Under Maintenance / Some criteria not fulfilled.
- **Red:** Not Operational / Tool Down / Critical Vulnerability.

For example, a Green indication under “**Security**” signals that the tool fulfills security criteria (Low severity), whereas a Red light under “**Privacy**” indicates unfulfilled privacy criteria. A “**Help**” button

(Feature L) opens a pop-up (**Figure 11**) explaining the specific meaning of the Green, Orange, and Red indications for operational, security, and privacy states .

6 Integration and Interoperability

6.1 Backend–frontend integration

This chapter describes the secure client-server communication model established by SIMAVI, utilizing a robust security layer enforced by Keycloak.

- **Technology Stack:** The solution is built on a modern stack comprising **Angular** for the frontend, **Java/Spring Boot** for the backend, and **Keycloak** for Identity and Access Management (IAM).
- **Authentication Flow (SSO):** The Angular application initializes Keycloak using the `keycloak-angular` library, setting the initial load option to `onLoad: 'check-sso'`. This configuration ensures seamless Single Sign-On (SSO) by relying on the user being authenticated against the **same Keycloak realm** used by the backend.
- **Security Layer (Token Forwarding):** All communication between the frontend and the backend is secured via an **Angular Interceptor**. This mechanism automatically intercepts every outgoing HTTP request and attaches the valid Keycloak Access Token (JWT) to the header in the format: `Authorization: Bearer <token>`.
- **Access Control:** User roles (e.g., `user`, `moderator`) are read directly from the Keycloak token payload. The Angular application uses this role information to enforce authorization at the UI level (e.g., showing or hiding buttons such as "Create new topic").

6.2 API specifications and communication flows

This section details the service-to-service communication structure and the critical token validation process.

- **Token Validation (Crucial):** Upon receiving a request from the frontend, the Java Backend validates the Keycloak Access Token (JWT). The backend uses the **Keycloak public key** (via a resource server adapter) to verify the token's signature, ensuring it is valid, unexpired, and issued by the trusted realm before processing the request.
- **Service-to-Service Communication (KT):** The **SIMAVI communication layer** acts as a proxy, calling the external Knowledge Tools (KT) services using **REST/HTTP**. These internal KT services handle AI processing tasks, including:
 - `/api/image-check/`
 - `/api/text-check/`
 - `/api/translate-text/`
 - `/api/summarize-text/`
- **Internal Networking (Security):** The communication layer calls the backend layer, which handles database operations (read/write data to MongoDB). The backend component

cannot be accessed outside of the internal environment, ensuring that all data manipulation is protected by the validated communication layer.

- **API Documentation:** All communication layer endpoints are documented using **Swagger** (OpenAPI specification).

The full API Specification can be found in Annex section 10.1.2

6.3 Data storage and retrieval mechanisms

This chapter outlines the persistence strategy and data access framework.

- **Database Technology:** The platform uses **MongoDB**, a NoSQL document store, for persistent data storage.
- **Data Access Layer (DAL):** Data interaction is managed by **Spring Data MongoDB**. The application uses the **MongoTemplate** for complex data operations, such as query execution and aggregation (e.g., for calculating user rankings).
- **Data Structure Strategy:** Entity relationships are established through explicit references:
 - The **Proposal** entity uses the **issueId** field to link to its parent **Topic**.
 - The **Argument** entity uses the **proposalId** field to link to its parent **Proposal**.

7 Testing and Validation in Pilots

This chapter describes the **testing and validation activities** carried out for the final ITHACA platform prototype. Given the prototype-oriented nature of the platform, the emphasis is placed on **technical verification, functional validation, and iterative refinement based on pilot feedback**, rather than on large-scale performance benchmarking or long-term operational validation.

Testing and validation activities were conducted incrementally throughout the development lifecycle and intensified during the pilot phase, ensuring that the platform reached a stable, usable, and demonstrable state aligned with project objectives.

7.1 Technical Testing of the Prototype

Technical testing focused on verifying the **correct integration, stability, and interoperability** of the platform's core components within the final deployment environment.

The main areas covered by technical testing included:

- **Deployment and infrastructure testing**, validating container startup, service dependencies, networking, and secure access via the reverse proxy.
- **Backend–frontend integration**, ensuring correct API communication, authentication flows, and data exchange.
- **Authentication and authorisation**, testing user registration, login, role-based access control, and token validation via Keycloak.
- **AI service integration**, confirming that AI-powered functionalities (summarisation, translation, toxicity detection) could be reliably invoked and that responses were correctly handled by the platform.
- **Error handling and logging**, verifying that failures or misconfigurations were traceable and did not compromise platform stability.

Testing was primarily performed through a combination of:

- Developer-led functional checks,
- Scenario-based testing of user flows,
- Controlled testing within the pilot infrastructure.

These activities ensured that the prototype was technically robust enough to support pilot operation and user validation.

7.2 Validation of Platform Functionalities

Validation activities focused on confirming that the implemented functionalities behaved as expected from a **user and stakeholder perspective**, in line with the design goals defined in WP3.

Key validated aspects included:

- **User onboarding and profile management**, including registration, login, and role differentiation.
- **Civic discussion workflows**, covering topic creation (moderators), proposal submission, argument posting, voting, and visualisation.
- **AI-assisted interaction**, ensuring that summarisation, translation and simplification features were accessible, understandable, and clearly marked as AI-generated.
- **Moderation mechanisms**, validating the interaction between automated toxicity detection and human moderator review.
- **Gamification features**, confirming the visibility and consistency of missions, experience points, badges, and leaderboards.

Validation was carried out through hands-on use of the platform during pilot activities and internal walkthroughs with consortium partners and pilot stakeholders.

7.3 Feedback Collected During Pilots

User and stakeholder feedback was collected during the pilot deployments through a combination of:

- Direct user interaction with the platform,
- Informal feedback sessions and discussions,
- Observations made by pilot organisers and technical partners.

Feedback focused primarily on:

- **Usability and clarity** of the user interface,
- **Comprehensibility of AI-assisted outputs**, particularly summaries and translations,
- **Perceived usefulness** of gamification elements,
- **Navigation and information overload** in discussion-heavy topics,
- **Transparency and trust**, especially regarding moderation and AI involvement.

Rather than formal surveys or quantitative metrics, feedback collection prioritised **qualitative insights** relevant to improving the prototype's usability and alignment with civic participation needs.

7.4 Refinements and Adjustments Applied

Based on the testing and feedback activities, a series of **incremental refinements** were applied to the platform prototype. These refinements aimed to improve stability, usability, and clarity without introducing major architectural changes.

Examples of refinements include:

- Minor adjustments to user interface elements to improve navigation and readability.
- Improved labelling and visibility of AI-generated content to reinforce transparency.
- Tuning of moderation thresholds and workflows to better support human-in-the-loop review.
- Bug fixes and performance optimisations identified during pilot usage.
- Clarifications in user flows related to proposal submission, voting, and gamification feedback.

These refinements reflect the **iterative nature of prototype development** and contribute to a more coherent and usable final platform, while acknowledging that further enhancements would be expected in a production-scale deployment.

7.5 Summary of Testing and Validation Outcomes

Overall, the testing and validation activities demonstrate that the ITHACA platform prototype:

- Is **technically stable and functionally integrated**,
- Supports the intended civic participation workflows,
- Provides usable and transparent AI-assisted features,
- Can be effectively deployed and operated in pilot environments.

The final prototype achieves the maturity required for **real-world demonstration, evaluation, and future scaling considerations**, fulfilling the objectives of WP3.

8 Known Limitations and Future Enhancements

This chapter outlines the **current limitations** of the ITHACA platform prototype as implemented under WP3, as well as **planned improvements** identified through development experience and pilot validation. These limitations should be understood in the context of a **final research and demonstration prototype**, rather than a fully industrialised civic participation system.

8.1 Current limitations

Despite reaching a stable and functional state, the ITHACA platform prototype exhibits a number of limitations inherent to its scope, development timeframe, and research-oriented objectives.

Prototype Scope and Scale

The platform has been validated in controlled pilot environments with a limited number of users and municipalities. As a result, aspects such as large-scale concurrency, long-term operational performance, and high-traffic stress conditions have not been exhaustively tested.

AI Functional Boundaries

AI-powered features (e.g., summarisation, translation, toxicity detection) are designed to be assistive and conservative. While this supports trust and safety, it may result in:

- Occasional loss of nuance in summaries or translations,
- Conservative moderation thresholds that require frequent human review.

These limitations reflect deliberate design choices aligned with ethical and trustworthy AI principles.

Configurability and Local Adaptation

Some platform parameters (e.g. moderation thresholds, gamification rules, discussion timelines) are currently managed at configuration or administrative level. Broader self-service configurability for municipalities or administrators is limited in the current prototype.

User Interface and Accessibility Refinement

While the platform supports core accessibility features and multilingual interaction, additional refinements—such as advanced accessibility customisation, enhanced mobile optimisation, and broader assistive technology support—remain outside the scope of the current implementation.

Integration and Extensibility

The prototype focuses on internal platform workflows and selected external AI services. Integration with external civic systems (e.g. municipal open data portals, e-government services) has not been fully explored.

8.2 Planned improvements beyond the project lifetime

Based on pilot experience, stakeholder feedback, and technical evaluation, several **planned improvements** have been identified. These improvements indicate logical next steps for further development rather than guaranteed future implementation.

Scalability and Performance Enhancements

Future work could include stress testing and optimisation for larger user bases, as well as deployment on scalable orchestration environments (e.g., Kubernetes) to support broader adoption.

Refinement of AI-Assisted Features

Planned enhancements include:

- Improved summarisation quality for long and highly diverse discussions,
- Expanded language coverage and improved translation accuracy,
- Fine-tuning of toxicity detection thresholds based on local civic norms.

Enhanced Administrative and Moderation Tools

Additional tools could be introduced to support moderators and administrators, including:

- More advanced dashboards for monitoring participation and moderation activity,
- Configurable moderation and gamification parameters,
- Improved reporting and audit capabilities.

Extended Transparency and User Control

The Public AI Register and PIMS components could be further enriched with:

- More detailed explanations of AI behaviour and limitations,
- Enhanced user-facing controls for data and consent management,
- Expanded transparency features aligned with future AI regulatory requirements.

Broader Integration and Sustainability

Future iterations could explore tighter integration with municipal systems, open data platforms, and long-term hosting strategies, supporting sustainability beyond the project lifetime.

9 Conclusions

This deliverable has presented the **final prototype of the ITHACA platform**, developed within Work Package 3, documenting its architecture, deployment, functionalities, user interfaces, and validation activities. Building on the initial prototype described in Deliverable D3.2, the platform has matured into a **coherent, integrated, and pilot-tested system** that demonstrates how artificial intelligence can support inclusive, transparent, and trustworthy civic participation.

The final ITHACA platform brings together a set of interoperable components that support the full civic engagement lifecycle, including secure user management, structured discussion through topics, proposals and arguments, AI-assisted interaction, human-in-the-loop moderation, gamification-based engagement incentives, and transparency mechanisms such as the Public AI Register and Personal Information Management System. These elements have been implemented using a modular, container-based architecture, enabling flexible deployment and iterative refinement within real pilot environments.

Testing and validation activities confirmed that the platform is **technically stable, functionally complete, and usable** for its intended demonstration purposes. Through pilot deployments, the consortium validated key interaction flows, AI-assisted features, and moderation mechanisms, while collecting qualitative feedback from users and stakeholders. This feedback informed a series of targeted refinements, reinforcing usability, clarity, and trust without compromising ethical or legal safeguards.

The platform's AI-powered functionalities—summarisation, translation, simplification, and toxicity detection—were intentionally designed as **assistive tools**, enhancing accessibility and comprehension while preserving human agency and oversight. This approach reflects the project's commitment to trustworthy AI, transparency, and compliance with data protection and ethical requirements, as reinforced through alignment with the Data Management Plan and related impact assessments.

At the same time, the deliverable acknowledges the **prototype nature** of the platform. Certain limitations remain in terms of scalability, configurability, and long-term operational readiness. These limitations are not shortcomings, but rather a natural outcome of a research and innovation project focused on validation and demonstration. Clear pathways for future improvements have been identified, outlining how the platform could evolve into a production-grade solution with broader municipal adoption.

In conclusion, Deliverable D3.4 demonstrates that the objectives of WP3 have been successfully achieved. The ITHACA platform final prototype provides a **robust, transparent, and ethically grounded foundation** for AI-supported civic participation, offering valuable insights and technical assets that can inform future research, policy experimentation, and real-world deployment beyond the project's lifetime.

10 Annexes

10.1.1 Deployment configuration files (Docker Compose, Kubernetes manifests)

Ngix config for ithaca.simavi.ro:

```
server {
    root /var/www/ithaca.simavi.ro/html;
    index index.html index.htm index.nginx-debian.html;
    server_name ithaca.simavi.ro www.ithaca.simavi.ro;
    if ($request_method = 'OPTIONS') {
        return 204;
    }
    location / {
        proxy_pass http://172.18.0.1:4200;
        proxy_redirect default;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    location /.well-known/acme-challenge/ {
        root /var/www/ithaca.simavi.ro;
    }

    location ^~ /api/combackend/ {
        proxy_pass http://172.18.0.1:8090;
        proxy_redirect default;
    }

    location ^~ /api/keycloak/ {
        proxy_pass http://10.222.30.234:8080;
        proxy_redirect default;
    }
}
```

```

# sockjs
location /api/sockjs {
    proxy_pass http://10.222.30.234:8090/;
    proxy_http_version 1.1;
    proxy_redirect off;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
}

location ^~ /api/broker/ {
    proxy_pass http://10.222.30.106:8090/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade websocket;
    proxy_set_header Connection upgrade;
}

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/ithaca.simavi.ro/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/ithaca.simavi.ro/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = ithaca.simavi.ro) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name ithaca.simavi.ro www.ithaca.simavi.ro;

    return 404; # managed by Certbot
}

```

Main docker-compose:

version: '3.8'

services:

ithaca-persistence:

build: ./backend-persistence

container_name: ithaca-persistence

ports:

- "8070:8070"

- "5005:5005"

environment:

SPRING_DATASOURCE_URL: jdbc:postgresql://10.222.30.234:5432/ithaca

SPRING_DATASOURCE_USERNAME: postgres

SPRING_DATASOURCE_PASSWORD: redacted

SPRING_PROFILES_ACTIVE: platform

volumes:

- /home/ithaca/pictures:/home/ithaca/pictures

depends_on:

- postgres

networks:

- app-network

ithaca-communication:

build: ./backend-communication

container_name: ithaca-communication

ports:

- "8090:8090"

- "5006:5006"

environment:

SPRING_PROFILES_ACTIVE: platform

networks:

- app-network

ithaca-orchestrator:

```
build: ./backend-orchestrator
container_name: ithaca-orchestrator
ports:
  - "8071:8071"
  - "5007:5007"
environment:
  SPRING_PROFILES_ACTIVE: platform
volumes:
  - /home/ithaca/test/python-app:/app/python-app
networks:
  - app-network
```

```
ithaca-audit:
  build: ./backend-audit
  container_name: ithaca-audit
  ports:
    - "8081:8081"
    - "5009:5009"
  environment:
    SPRING_PROFILES_ACTIVE: platform
  networks:
    - app-network
```

```
postgres:
  image: postgres:latest
  container_name: postgresdb
  ports:
    - "5432:5432"
  networks:
    - app-network
  environment:
    POSTGRES_DB: ithaca
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: redacted
```

volumes:

- postgres-data:/var/lib/postgresql/data

ithaca-app:

image: ithaca-app

container_name: ithaca-frontend

build:

context: ./frontend

ports:

- "4200:4200"

networks:

- app-network

volumes:

mongo-data:

postgres-data:

pgadmin-data:

networks:

app-network:

external:

name: app-network

Frontend dockerfile:

Use Node.js as the base image

FROM node:16-alpine

Set the working directory in the container

WORKDIR /usr/src/app

Copy the Angular dist folder into the container

COPY ./ITHACA-frontend /usr/src/app/dist

Install a lightweight HTTP server (e.g., serve)

RUN npm install -g serve

Expose port 4200

EXPOSE 4200

```
# Serve the Angular app
```

```
CMD ["serve", "-s", "/usr/src/app/dist", "-l", "4200"]
```

Ithaca-Communication dockerfile

```
# Use an official OpenJDK runtime as a parent image
```

```
FROM bellsoft/liberica-openjdk-debian:17
```

```
# Set the working directory in the container
```

```
WORKDIR /app
```

```
# Copy the JAR file into the container
```

```
COPY ithaca-0.0.1-SNAPSHOT.jar /app/app.jar
```

```
# Expose both the application and debug ports
```

```
EXPOSE 8090 5006
```

```
# Run the JAR file with remote debugging enabled
```

```
ENTRYPOINT ["java", "-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=0.0.0.0:5006", "-jar", "/app/app.jar"]
```

Ithaca-Persistence dockerfile:

```
# Use an official OpenJDK runtime as a parent image
```

```
FROM bellsoft/liberica-openjdk-debian:17
```

```
# Set the working directory in the container
```

```
WORKDIR /app
```

```
# Copy the jar file into the container
```

```
COPY ITHACA-0.0.1-SNAPSHOT.jar /app/app.jar
```

```
# Expose both the application and debug ports
```

```
EXPOSE 8070 5005
```

```
# Run the jar file with remote debugging enabled
```

```
ENTRYPOINT ["java", "-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=0.0.0.0:5005", "-jar", "/app/app.jar"]
```

Ithaca-Orchestrator dockerfile:

```
# Use an official OpenJDK runtime as a parent image
```

```
FROM bellsoft/liberica-openjdk-debian:17
```

```
# Set the working directory
```

```
WORKDIR /app
```

```
# Install Python & dependencies
```

```
RUN apt-get update && \  
  apt-get install -y python3 python3-pip python3-venv && \  
  apt-get clean  
# Ensure the python-app directory exists  
RUN mkdir -p /app/python-app  
# Create a virtual environment  
RUN python3 -m venv /app/python-app/venv  
# Activate venv & install requirements  
COPY requirements.txt /app/python-app/requirements.txt  
RUN /app/python-app/venv/bin/pip install --no-cache-dir -r /app/python-app/requirements.txt  
# Copy the JAR file  
COPY ithaca-orchestrator-0.0.1-SNAPSHOT.jar /app/app.jar  
# Expose ports  
EXPOSE 8071 5007  
# Run the application  
ENTRYPOINT ["java", "-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=0.0.0.0:5007", "-jar", "/app/app.jar"]
```

Ithaca-audit dockerfile:

```
# Use an official OpenJDK runtime as a parent image  
FROM bellsoft/liberica-openjdk-debian:17  
# Set the working directory in the container  
WORKDIR /app  
# Copy the jar file into the container  
COPY audit-0.0.1-SNAPSHOT.jar /app/app.jar  
# Expose both the application and debug ports  
EXPOSE 8081 5009  
# Run the jar file with remote debugging enabled  
ENTRYPOINT ["java", "-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=0.0.0.0:5009", "-jar", "/app/app.jar"]
```

Keycloak docker-compose:

```
services:  
  keycloak_auth:
```

image: quay.io/keycloak/keycloak:24.0.1

container_name: keycloak_auth

environment:

KC_DB: postgres

KC_DB_URL: jdbc:postgresql://10.222.30.234:5432/ithaca

KC_DB_USERNAME: postgres

KC_DB_PASSWORD: redacted

#KC_HOSTNAME: keycloak.ithaca.simavi.ro

#KC_HOSTNAME_PORT: 80

#KC_HOSTNAME_STRICT-BACKCHANNEL: 'true'

#KC_HOSTNAME_STRICT: 'false'

#KC_HOSTNAME_STRICT_HTTPS: 'false'

##KC_HOSTNAME_STRICT_HTTPS: 'true'

KC_LOG_LEVEL: info

#KC_METRICS_ENABLED: 'true'

#KC_HEALTH_ENABLED: 'true'

KEYCLOAK_ADMIN: admin

KEYCLOAK_ADMIN_PASSWORD: redacted

KC_HTTPS_CERTIFICATE_FILE: /opt/keycloak/certs/fullchain.pem

KC_HTTPS_CERTIFICATE_KEY_FILE: /opt/keycloak/certs/privkey.pem

KC_HOSTNAME_STRICT_HTTPS: "true"

KC_PROXY: edge

command: start-dev --verbose

extra_hosts:

- "keycloak.ithaca.simavi.ro:10.222.30.234"

networks:

- app-network

ports:

- 8080:8080

volumes:

- ./ithaca-dashboard:/opt/keycloak/themes/ithaca-dashboard/
- ./custom-ithaca:/opt/keycloak/themes/custom-ithaca/
- /etc/letsencrypt/live/keycloak.ithaca.simavi.ro/fullchain.pem:/opt/keycloak/certs/fullchain.pem
- /etc/letsencrypt/live/keycloak.ithaca.simavi.ro/privkey.pem:/opt/keycloak/certs/privkey.pem

networks:

app-network:

external:

name: app-network

MongoDB docker-compose:

version: '3.8'

services:

mongo:

image: mongo:latest

container_name: mongoddb

restart: always

ports:

- "27017:27017"

networks:

- app-network

volumes:

- mongo-data:/data/db

environment:

MONGO_INITDB_ROOT_USERNAME: root

MONGO_INITDB_ROOT_PASSWORD: redacted

command: mongod --bind_ip_all --port 27017 --auth

volumes:

mongo-data:

networks:

app-network:

external:

name: app-network

Gamification docker-compose:

version: '3.8'

services:

upat-aifairness:

build: .

container_name: upat-gamification

ports:

- "8015:8015"

environment:

- PYTHONUNBUFFERED=1

restart: unless-stopped

networks:

- app-network

networks:

app-network:

external: true

name: app-network

Gamification dockerfile:

Use official lightweight Python image

FROM python:3.11-slim

Set environment vars

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

Set work directory

WORKDIR /app

Install system deps (needed for numpy)

```
RUN apt-get update && apt-get install -y \
```

```
  build-essential \
```

```
  && rm -rf /var/lib/apt/lists/*
```

```
# Copy requirements first (for caching)
```

```
COPY requirements.txt .
```

```
# Install dependencies
```

```
RUN pip install --no-cache-dir -r requirements.txt
```

```
# Copy project files
```

```
COPY . .
```

```
# Expose port
```

```
EXPOSE 8015
```

```
# Run app with uvicorn
```

```
CMD ["uvicorn", "gamification:app", "--host", "0.0.0.0", "--port", "8015"]
```

```
Gamification requirements.txt:
```

```
fastapi
```

```
uvicorn[standard]
```

```
numpy
```

```
pydantic
```

```
AI-Fairness docker-compose:
```

```
version: '3.8'
```

```
services:
```

```
  upat-aifairness:
```

```
    build: .
```

```
    container_name: upat-aifairness
```

```
    ports:
```

```
      - "8011:8011"
```

```
    environment:
```

```
      - PYTHONUNBUFFERED=1
```

```
restart: unless-stopped
```

```
networks:
```

```
- app-network
```

```
networks:
```

```
  app-network:
```

```
    external:
```

```
      name: app-network
```

AI-Fairness dockerfile:

```
# Use a lightweight Python image
```

```
FROM python:3.11-slim
```

```
# Set the working directory inside the container
```

```
WORKDIR /app
```

```
# Copy the requirements file first (better for caching layers)
```

```
COPY requirements.txt .
```

```
# Install dependencies
```

```
RUN pip install --no-cache-dir -r requirements.txt
```

```
RUN pip install 'aif360[Reductions]'
```

```
RUN pip install 'aif360[inFairness]'
```

```
RUN pip install 'aif360[OptimalTransport]'
```

```
RUN pip install 'aif360[AdversarialDebiasing]'
```

```
# Copy the rest of the application code
```

```
COPY . .
```

```
# Expose the FastAPI port
```

```
EXPOSE 8011
```

```
# Command to run the FastAPI app with Uvicorn
```

```
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8011"]
```

```
main.py:
```

```
from fastapi import FastAPI
from pydantic import BaseModel, Field
import numpy as np
import json # Import the json module to convert Python objects to JSON strings
from evaluate_fairness_func import evaluate_fairness, json_to_array # Import functions

app = FastAPI()

# Define the request data structure
class RequestData(BaseModel):
    isToxic: bool = Field(..., alias="toxic")
    isVulnerable: bool = Field(..., alias="vulnerable")
    isPositive: bool = Field(..., alias="positive")

@app.post("/evaluate_fairness")
async def evaluate_fairness_endpoint(request: list[RequestData]): #  Expecting JSON array
    try:
        json_string = json.dumps([r.dict() for r in request])
        json_string = json_string.replace("'", "")
        # Convert the JSON string to numpy arrays
        y_pred, val_labels_plus_vul = json_to_array(json_string)

        # Call the fairness evaluation function
        fairness_metrics = evaluate_fairness(y_pred, val_labels_plus_vul)
        fairness_metrics = fairness_metrics.replace('Infinity', "Infinity").replace('NaN', 'null')
        fairness_metrics_dict = json.loads(fairness_metrics)

        return fairness_metrics_dict["Comment0"]

    except Exception as e:
        raise HTTPException(status_code=400, detail=f"Error evaluating fairness: {str(e)}")
```

AI-Fairness requirements.txt

```
fastapi
uvicorn
```

aif360==0.6.0

matplotlib==3.8.4

numpy==2.2.4

pandas==2.2.3

10.1.2 API documentation

Topics API Specification:

topic-controller		^
GET	/api/topics	∨
PUT	/api/topics	∨
POST	/api/topics	∨
POST	/api/topics/{id}/vote	∨
POST	/api/topics/{id}/incrementViews	∨
POST	/api/topics/unfeatured	∨
POST	/api/topics/makeFeature	∨
POST	/api/topics/incrementViews2	∨
POST	/api/topics/featured	∨
GET	/api/topics/{id}	∨
GET	/api/topics/count	∨
DELETE	/api/topics/{topicId}	∨
DELETE	/api/topics/deleteTopicByIdWithAllTheContent/{topicId}	∨

Proposals API Specification:

issues-controller ^

POST	/api/proposals	∨
POST	/api/proposals/{proposalId}/vote	∨
GET	/api/issues	∨
POST	/api/issues	∨
GET	/api/proposals/{issueId}	∨
GET	/api/proposals/proposal/{proposalId}	∨
GET	/api/issues/{id}	∨
DELETE	/api/proposals/deleteProposalAndAllTheArguments/{proposalId}	∨

Arguments API Specification:

argument-controller ^

PUT	/api/arguments/{argumentId}/clear-reports	∨
POST	/api/{argumentId}/report	∨
POST	/api/arguments	∨
POST	/api/arguments/vote	∨
POST	/api/arguments/editComment	∨
GET	/api/arguments/{proposalId}	∨
GET	/api/arguments/toxic	∨
DELETE	/api/arguments/deleteById/{argumentId}	∨

Profile API Specification:

profile-controller

POST	/api/profile/upload-profile-picture	📱	⌵
POST	/api/profile/update/missionData	📱	⌵
POST	/api/profile/update/language		⌵
POST	/api/profile/details		⌵
GET	/api/profile/email		⌵

KT API Specification:

kt-controller

POST	/api/kt/translate-text	⌵
POST	/api/kt/text-check	⌵
POST	/api/kt/summarize-check	⌵
POST	/api/kt/image-check	⌵

Poll API Specification:

poll-controller

GET	/api/polls	⌵
POST	/api/polls	⌵
POST	/api/polls/vote	⌵
GET	/api/polls/results/{pollId}	⌵
GET	/api/polls/new	⌵

Login Logs API Specification:

login-controller

POST	/api/login/log-ip	⌵
GET	/api/login/login-logs	⌵

Log actions API Specification:

log-actions-controller ^

GET	/api/log-actions	∨
POST	/api/log-actions	∨
GET	/api/log-actions/time	∨
GET	/api/log-actions/error	∨

Keycloak User API Specification:

keycloak-user-controller ^

POST	/api/keycloak/friends/remove	∨
POST	/api/keycloak/friends/add	∨
GET	/api/keycloak/users	∨
GET	/api/keycloak/stats	∨
GET	/api/keycloak/friends/{keycloakId}	∨

Feedback API Specification:

feedback-controller ^

GET	/api/feedback	∨
POST	/api/feedback	∨
DELETE	/api/feedback/{id}	∨

Audit API Specification:

audit-controller ^

GET	/api/audit	∨
POST	/api/audit	∨

UPAT API Specification:

ai-tools-controller		^
POST	/api/ai/upat-fairness	∨
GET	/api/ai/tools-status	∨
GET	/api/ai/security-levels	∨
GET	/api/ai/leaderboard/{location}	∨

10.1.3 Code samples for AI-powered toolkit

10.1.3.1 Language detection

```
def detect_language(text):
    # Language prediction model

    model_path = hf_hub_download(repo_id="facebook/fasttext-language-
identification", filename="model.bin")

    model = fasttext.load_model(model_path)
    prediction = model.predict(text)
    lang = prediction[0][0].replace("__label__", "")
    # Convert prediction to meaningful language code (sk, ro, en etc.):
    lang = langcodes.Language.get(lang)
    confidence = prediction[1][0]
    return lang.language, confidence
```

10.1.3.2 Translation

```
class TranslateTextView(views.APIView):
    def post(self, request, format=None):
        # Models to use assuming that target language is English:
        models_from = {
            'ro': "Helsinki-NLP/opus-mt-ROMANCE-en", # "Helsinki-NLP/opus-
mt-ro-en"
            'sk': "Helsinki-NLP/opus-mt-sk-en"
        }
        # Models to use assuming that language of origin is English:
        models_to = {
            'ro': "Helsinki-NLP/opus-mt-en-ro",
            'sk': "Helsinki-NLP/opus-mt-en-sk"
        }
```

```

if request.data['from'] == 'en':
    model_name = models_to[request.data['to']]
else:
    model_name = models_from[request.data['from']]

    tokenizer = MarianTokenizer.from_pretrained(model_name,
token=settings.HUGGING_TOKEN)
    model = MarianMTModel.from_pretrained(model_name,
token=settings.HUGGING_TOKEN)

    translated_text = translate_text(request.data['text'], tokenizer,
model)

    return Response({'translation':translated_text},
status=status.HTTP_200_OK)

def translate_text(text, tokenizer, model):
    inputs = tokenizer(text, return_tensors="pt", padding=True)
    translated = model.generate(**inputs)
    return tokenizer.decode(translated[0], skip_special_tokens=True)

```

10.1.3.3 Summarization

```

def summarize(text):
    # Load a tokenizer and a summarization model:
    model_name = "facebook/bart-large-cnn"
    tokenizer = AutoTokenizer.from_pretrained(model_name,
token=settings.HUGGING_TOKEN)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name,
token=settings.HUGGING_TOKEN)
    # Create a summarization pipeline:
    summarizer = pipeline("summarization", model=model,
tokenizer=tokenizer)

    # max_length and min_length are number of tokens of the summarized text.
    # These numbers can be updated accordingly after observing results.
    summary = summarizer(text, max_length=100, min_length=30,
do_sample=False)
    # Return summarized text:
    return summary[0]['summary_text']

```

10.1.3.4 Categorization and clustering

```

from keybert import KeyBERT
from sentence_transformers import SentenceTransformer
from collections import defaultdict

embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
kw_model = KeyBERT(embedding_model)

def extract_keyphrases(text, top_n=10):
    keywords = kw_model.extract_keywords(
        text,
        keyphrase_ngram_range=(1, 3),
        stop_words="english",
        top_n=top_n
    )
    return [kw for kw, score in keywords]

def aggregate_terms(corpus):
    term_stats = defaultdict(lambda: {
        "frequency": 0,
        "sources": set(),
        "languages": set()
    })

    for item in corpus:
        phrases = extract_keyphrases(item["text"])
        for phrase in phrases:
            key = phrase.lower()
            term_stats[key]["frequency"] += 1
            term_stats[key]["sources"].add(item["source"])
            term_stats[key]["languages"].add(item["language"])

    return term_stats

from sklearn.cluster import AgglomerativeClustering
import numpy as np

```

```
def cluster_terms(terms):
    embeddings = embedding_model.encode(terms)
    clustering = AgglomerativeClustering(
        n_clusters=None,
        distance_threshold=1.2
    )
    labels = clustering.fit_predict(embeddings)

    clusters = defaultdict(list)
    for term, label in zip(terms, labels):
        clusters[label].append(term)

    return clusters
```

10.1.3.4 Categorization and clustering

10.1.4 Vocabulary and Ontology

10.4.1.1 Keyphrase extraction

```
from keybert import KeyBERT
from sentence_transformers import SentenceTransformer
from collections import defaultdict
import json

# Multilingual embedding model
embedding_model = SentenceTransformer(
    "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
)

kw_model = KeyBERT(model=embedding_model)

def extract_keyphrases(
    documents,
    language,
    top_n=15
):
```

```

all_phrases = defaultdict(int)

for doc in documents:
    keywords = kw_model.extract_keywords(
        doc,
        keyphrase_ngram_range=(1, 3),
        stop_words=language,
        top_n=top_n
    )
    for phrase, score in keywords:
        all_phrases[phrase.lower()] += 1

return sorted(
    all_phrases.items(),
    key=lambda x: x[1],
    reverse=True
)

if __name__ == "__main__":
    with open("municipal_articles.json") as f:
        data = json.load(f)

    results = {}

    for lang, docs in data.items():
        results[lang] = extract_keyphrases(
            documents=docs,
            language=lang
        )

    with open("candidate_terms.json", "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)

```

10.1.4.2 Starter vocabulary

```

{
  "metadata": {
    "version": "1.0",
    "generation_method": "automatic_extraction",

```

```

"sources": [
  "municipal_websites",
  "news_articles"
],
"languages": ["sk", "ro"],
"pivot_language": "en"
},
"terms": [
  {
    "id": "term_001",
    "label": "winter maintenance",
    "language": "en",
    "source_language": "sk",
    "original_forms": ["zimná údržba"],
    "frequency": 14
  },
  {
    "id": "term_002",
    "label": "waste collection",
    "language": "en",
    "source_language": "sk",
    "original_forms": ["zber odpadu", "komunálny odpad"],
    "frequency": 19
  },
  {
    "id": "term_003",
    "label": "district heating",
    "language": "en",
    "source_language": "sk",
    "original_forms": ["dodávka tepla", "teplá voda"],
    "frequency": 11
  },
  {
    "id": "term_004",
    "label": "public announcement",
    "language": "en",
    "source_language": "ro",
    "original_forms": ["anunț public"],
    "frequency": 22
  },
  {
    "id": "term_005",
    "label": "municipal property rental",
    "language": "en",
    "source_language": "sk",
    "original_forms": ["prenájom majetku mesta"],
    "frequency": 7
  },
  {
    "id": "term_006",
    "label": "citizen services",
    "language": "en",
    "source_language": "ro",
    "original_forms": ["servicii pentru cetățeni"],

```

```

    "frequency": 16
  },
  {
    "id": "term_007",
    "label": "road closure",
    "language": "en",
    "source_language": "ro",
    "original_forms": ["închidere drum"],
    "frequency": 9
  },
  {
    "id": "term_008",
    "label": "public event",
    "language": "en",
    "source_language": "sk",
    "original_forms": ["verejné podujatie"],
    "frequency": 12
  }
]
}

```

10.1.4.3 Normalized - semantically structured vocabulary

```

{
  "metadata": {
    "version": "1.1",
    "curation": "semi-automatic",
    "pivot_language": "en",
    "aligned_with": [
      "e-government",
      "municipal_services",
      "civic_participation"
    ]
  },
  "concepts": [
    {
      "concept_id": "C001",
      "preferred_label": "Municipal Services",
      "alt_labels": [
        "Citizen services",
        "Public services"
      ],
      "definition": "Services provided by a municipality to citizens.",
      "broader": null,
      "narrower": [
        "C002",
        "C003",
        "C004"
      ],
      "mapped_terms": ["term_006"]
    },
    {
      "concept_id": "C002",

```

```

    "preferred_label": "Waste Management",
    "alt_labels": [
      "Waste collection",
      "Municipal waste"
    ],
    "definition": "Collection and processing of household and public waste.",
    "broader": "C001",
    "narrower": [],
    "mapped_terms": ["term_002"]
  },
  {
    "concept_id": "C003",
    "preferred_label": "Heating Services",
    "alt_labels": [
      "District heating",
      "Hot water supply"
    ],
    "definition": "Provision of heating and hot water by municipal systems.",
    "broader": "C001",
    "narrower": [],
    "mapped_terms": ["term_003"]
  },
  {
    "concept_id": "C004",
    "preferred_label": "Infrastructure Maintenance",
    "alt_labels": [
      "Winter maintenance",
      "Road maintenance"
    ],
    "definition": "Maintenance of public infrastructure and roads.",
    "broader": "C001",
    "narrower": [],
    "mapped_terms": ["term_001"]
  },
  {
    "concept_id": "C005",
    "preferred_label": "Public Communication",
    "alt_labels": [
      "Public announcements",
      "Municipal notices"
    ],
    "definition": "Official communication issued by municipal authorities.",
    "broader": null,
    "narrower": [],
    "mapped_terms": ["term_004"]
  },
  {
    "concept_id": "C006",
    "preferred_label": "Civic Life",
    "alt_labels": [
      "Public events",
      "Community activities"
    ],
    "definition": "Events and activities organized for citizens.",

```

```
    "broader": null,  
    "narrower": [],  
    "mapped_terms": ["term_008"]  
  }  
]  
}
```

10.1.4.4 Vocabulary to ontology

```
from rdflib import Graph, Literal, Namespace, RDF  
from rdflib.namespace import SKOS  
  
import json  
import uuid  
  
EX = Namespace("http://example.org/municipal-vocab/")  
  
g = Graph()  
g.bind("skos", SKOS)  
g.bind("ex", EX)  
  
with open("../vocabulary/validated_terms.json") as f:  
    terms = json.load(f)  
  
for term in terms:  
    concept_uri = EX[str(uuid.uuid4())]  
  
    g.add((concept_uri, RDF.type, SKOS.Concept))  
    g.add((concept_uri, SKOS.prefLabel,  
          Literal(term["label_en"], lang="en")))  
    g.add((concept_uri, SKOS.altLabel,  
          Literal(term["label_original"], lang=term["language"])))  
  
    g.add((concept_uri, SKOS.note,  
          Literal(f"Frequency: {term['frequency']}")))  
  
g.serialize(  
    destination="municipal_vocab.skos.ttl",  
    format="turtle"  
)
```

10.1.4.5 Ontology Representation

```
@prefix ex: <http://example.org/municipal#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:MunicipalServices a skos:Concept ;
    skos:prefLabel "Municipal Services"@en ;
    skos:altLabel "Citizen services"@en ;
    skos:definition "Services provided by a municipality to citizens."@en ;
    skos:narrower ex:WasteManagement ,
                 ex:HeatingServices ,
                 ex:InfrastructureMaintenance .

ex:WasteManagement a skos:Concept ;
    skos:prefLabel "Waste Management"@en ;
    skos:altLabel "Waste collection"@en ;
    skos:definition "Collection and processing of household and public waste."@en
;
    skos:broader ex:MunicipalServices .

ex:HeatingServices a skos:Concept ;
    skos:prefLabel "Heating Services"@en ;
    skos:altLabel "District heating"@en ;
    skos:definition "Provision of heating and hot water by municipal systems."@en
;
    skos:broader ex:MunicipalServices .

ex:InfrastructureMaintenance a skos:Concept ;
    skos:prefLabel "Infrastructure Maintenance"@en ;
    skos:altLabel "Winter maintenance"@en ;
    skos:definition "Maintenance of public infrastructure and roads."@en ;
    skos:broader ex:MunicipalServices .

ex:PublicCommunication a skos:Concept ;
    skos:prefLabel "Public Communication"@en ;
    skos:altLabel "Public announcements"@en ;
    skos:definition "Official communication issued by municipal authorities."@en
.

ex:CivicLife a skos:Concept ;
    skos:prefLabel "Civic Life"@en ;
    skos:altLabel "Public events"@en ;
    skos:definition "Events and activities organized for citizens."@en .
```