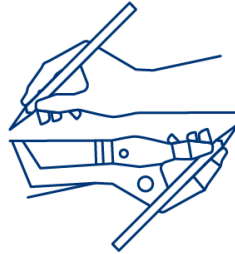




Funded by
the European Union

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

D3.1 ITHACA Technical requirements, architecture design and 1st release - report

Work Package 3: ITHACA platform design and development

Authors:	KONNEKTABLE
Status:	Final Version
Due Date:	31/01/25
Version:	1.0
Submission Date:	17/06/2025
Dissemination Level:	PU

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
MARTIN	MESTO MARTIN	SK

DOCUMENT HISTORY

VERSION	DATE	CHANGES	RESPONSIBLE PARTNER
0.1	11/03/2025		KONNEKTABLE
0.2	01/04/2025	Additions	UPAT and SIMAVI
0.3	15/05/2025	Additions	KONNEKTABLE
0.4	10/06/2025	Review and adaptations	KONNEKTABLE
Final	17/06/2025	Submission to EC	KONNEKTABLE

TABLE OF CONTENTS

TABLE OF CONTENTS	4
LIST OF FIGURES	7
LIST OF TABLES	7
ABBREVIATIONS	8
EXECUTIVE SUMMARY	9
1. Introduction to ITHACA	11
1.1 Background of the ITHACA Project	11
1.2 Purpose of Deliverable D3.1	11
1.3 Structure of the Document	12
2. Technical Requirements Analysis (T3.1)	13
2.2 Functional Requirements	13
2.2.1 User Registration, Authentication, and Roles	13
2.2.2 Civic Forum and Topic Discussions	14
2.2.3 Argument Visualization and Deliberation Tools	14
2.2.4 AI Tools and Transparency Mechanisms	14
2.2.5 Gamification and Motivation Mechanisms	14
2.2.6 Personal Information Management System (PIMS)	14
2.2.7 Moderation and Complaint Handling	15
2.3 Non-Functional Requirements	15
2.3.1 Legal and Ethical Compliance	15
2.3.2 Architecture, Integration, and Deployment	15
2.3.3 Performance and Scalability	15
2.3.4 Usability and Accessibility	15
3. System Architecture Design (T3.2)	16
3.1 Introduction	16
3.1.1 Purpose of the Platform	16
3.1.2 Target audience	18
3.2 Overview of Hybrid Architectures	19
3.2.1 Definition of Hybrid Architectures	19
3.2.2 Advantages and Challenges	20
3.3 Microservices Architecture	20

Key characteristics of Microservices include:	20
3.3.1 Service Independence	20
3.3.2. Scalability	20
3.3.3 Maintainability	20
Design Principles	21
3.3.4 Decentralized Data Management	21
3.3.5 API-Based Communication	21
Implementation Best Practices	21
3.3.6 Service Discovery	21
3.3.7 Containerization	21
3.3.8 Orchestration	21
3.3.9 Microservices Security Considerations	21
3.4 Event-Driven Messaging Architecture	22
The key concepts of Event-Driven Architecture are:	22
Events	22
Event Producers and Consumers	22
3.4.1 Event-Driven Design Patterns	22
Publish-Subscribe	22
Event Sourcing	23
3.4.2 Implementation Technologies	23
Message Brokers	23
Event Hubs	23
3.4.3 Event-Driven Microservices Communication	23
3.5 Integration of Microservices and Event-Driven Messaging	23
3.5.1 Rationale for Integration	23
Challenges and Solutions	24
3.5.2 Consistency and Transactionality	24
3.5.3 Event Schema Evolution	24
3.6 Design Patterns for Hybrid Architectures	24
3.6.1 Command Query Responsibility Segregation (CQRS)	24
3.6.2 Saga Pattern	24
3.7 End-to-End Architecture Definition	25
3.7.1 Introduction	25
3.7.2 Layers of the application	25
3.7.3 Semantic Layer architecture	26

3.7.4 Summary of Tools used in All Use Cases	27
3.7.5 Component Interaction and Data Flow	29
3.7.6 Implementation Guidelines	36
Steps for Implementing a Hybrid Architecture	36
Monitoring and Debugging Strategies	37
3.7.7 Conclusion	38
4. Evidence Extraction from Citizen Discussions (T3.3)	38
4.1. Overview	38
4.2. Content Collection Methods	38
4.2.1 Web Crawling	38
4.2.2 Social Media Data Acquisition	39
4.3. Multilingual Content Processing and Analysis	39
4.3.1 Assessing data directly from original language (Romanian, Slovak)	39
4.3.2 Translating original text to English and subsequently assessing data	40
4.4 Integration into Platform	40
5. Cognitive and Social Agents (T3.4)	40
5.1 Categorization and clustering of texts	41
5.1.1 Categorization through sequence classification	41
5.1.2 Clustering and keywords through a sentence transformer	41
5.2 Automated moderation	41
5.3 Vocabulary Dataset	42
6. Ontology (T3.5)	42
7. Gamification Incentive Mechanism (T3.6)	43
ad I.: Self-determination theory as underlying theoretical framework to foster intrinsic motivation	44
Conclusions:	45
ad II.: Best-practice examples of gamification mechanics that work and are intuitively understandable	45
Conclusions:	46
ad III.: Making the 'rules' on how to get what kind of badge explicit	46
ad IV.: Avoidance of undesirable side effects of the gamification approach	46
8. Public AI Register (T3.7)	49
8.1. Requirements Gathering	49
8.2. Information Architecture Design	49
8.3. Wireframing and Prototyping	50
8.4. Implementation Planning	50

8.5. Accessibility and Usability Testing	50
9. Argument Visualization (T3.8)	50
10. Trust and Security Infrastructure (T3.9)	53
10.1. Security Architecture and Trust Mechanisms	53
10.1.1 Authentication and Authorization	53
10.1.2 Endpoint Protection	53
10.1.3 API Security:	54
10.1.4 Trust Mechanisms	54
10.2 Adherence to WS-Security and Interoperability Standards	54
10.3 Glossary Development and Encryption Methods	55
11. Personal Information Management System (T3.10)	55
12. System Integration, Testing, and Deployment (T3.11)	55
Role of CI/CD Practices in Development	56
Scalability Features	56
Security Considerations	56
13. Conclusions	57
14. References	58

LIST OF FIGURES

Figure 1 Layers of the application	25
Figure 2 Semantic Layer architecture.....	26
Figure 3 The screen appears in the ITHACA user interface, upon pressing the “Arguments” tab.	51
Figure 4 A first design of the window appearing in the ITHACA user interface, upon selecting an issue from the “Argument” tab (Figure 3).....	52

LIST OF TABLES

Table 1 Summary of Tools used in All Use Cases.....	27
Table 2 List of activities (clustered into easy, medium, and difficult) that can be carried out (and thus, rewarded by badges) only a single time	47
Table 3 List of activities worthy of reward, their ‘difficulty level’, and the badges (or types of basic needs they should facilitate).....	47

ABBREVIATIONS

AI	Artificial Intelligence
GDPR	General Data Protection Regulation
HCI	Human-Computer Interaction
KPI	Key Performance Indicator
NGO	Non-Governmental Organization
STT	Speech-to-Text
TTS	Text-to-Speech
UAT	User Acceptance Testing
WP	Work Package
WCAG	Web Content Accessibility Guidelines

EXECUTIVE SUMMARY

Deliverable D3.1 – *ITHACA Technical Requirements, Architecture Design, and 1st Release* – presents the foundational framework for the development of the ITHACA platform: a modular, AI-powered digital infrastructure designed to enhance civic participation in local governance. Anchored in the principles of transparency, inclusivity, security, and user empowerment, this deliverable outlines the initial technical and architectural blueprints, alongside the first release of the system components developed under Work Package 3 (WP3).

The ITHACA platform is part of a broader Horizon Europe initiative to leverage artificial intelligence for democratic innovation. Its central goal is to empower citizens—including those from underrepresented and vulnerable communities—to actively engage in civic discourse, co-decision-making, and local policy design. This deliverable consolidates inputs from participatory design (WP1), co-creation and evaluation (WP2), and ethical/legal frameworks (WP5) to ensure that technology development is guided by real-world user needs and European values such as data protection, accessibility, and fairness.

The report opens with a comprehensive analysis of technical requirements (Chapter 2), capturing both functional and non-functional demands informed by GDPR, digital accessibility (WCAG), and system interoperability. It specifies capabilities such as multilingual discourse facilitation, sentiment analysis, secure identity management, participatory dashboards, AI transparency tools, and granular data consent management.

Chapter 3 details the system’s hybrid architecture—a synergy of microservices and event-driven messaging. This design ensures flexibility, scalability, and modular deployment. Key components include a semantic knowledge base (ontology), data ingestion pipelines, privacy-preserving AI tools, and a multi-layered dashboard environment for civic interactions.

Subsequent chapters present the development status and design strategies for core components:

- **Evidence Extraction (Chapter 4):** Collects and analyzes multilingual citizen discourse from online and social media sources in pilot cities (Brasov and Martin), using NLP and sentiment analysis to detect public concerns.
- **Cognitive and Social Agents (Chapter 5):** Implements AI agents for topic clustering, automated moderation, and hate speech detection, aiding human moderators while upholding ethical AI use.
- **Ontology Development (Chapter 6):** Constructs a semantic framework drawing on public sector vocabularies (e.g., CPSV, FOAF), enabling structured data indexing, annotation, and retrieval.
- **Gamification Incentives (Chapter 7):** Applies motivational design grounded in Self-Determination Theory to boost meaningful engagement, with reward mechanisms for competence, relatedness, and activity.
- **Public AI Register (Chapter 8):** A transparency layer detailing AI components in use, their logic, datasets, and governance considerations, in line with emerging AI regulations.
- **Argument Visualization (Chapter 9):** Offers citizens a visual, interactive interface to explore public debates, contribute proposals, and engage with pros/cons and vote-weighted feedback mechanisms.

- **Trust and Security Infrastructure (Chapter 10):** Outlines an RBAC-secured environment using encrypted communication and GDPR-aligned safeguards to ensure integrity, accountability, and user trust.
- **Personal Information Management System (Chapter 11):** Enables users to manage their data, revoke consents, and exercise rights under data protection law through an intuitive interface.
- **System Integration and Deployment (Chapter 12):** Describes the CI/CD approach to containerized deployment, scalable AI model integration, and continuous security validation using Docker, Kubernetes, and automated pipelines.

The deliverable concludes with a roadmap (Chapter 13) toward the next development iteration (D3.2 and D3.3), focusing on enhancing platform functionalities, refining ethical alignment, and deepening pilot engagement. This iterative development ensures ITHACA remains responsive to societal challenges, regulatory shifts, and user feedback, supporting the emergence of a truly participatory civic AI infrastructure.

1. Introduction to ITHACA

1.1 Background of the ITHACA Project

The ITHACA project, funded under the Horizon Europe programme, aims to enhance civic participation through the development and deployment of an AI-driven platform that empowers citizens to engage actively in democratic processes. The project's core objectives centre on creating a transparent, inclusive, and accessible platform for civic engagement, where citizens can voice their perspectives and collaborate with local governance entities in decision-making processes. ITHACA addresses contemporary challenges around democratic participation by implementing artificial intelligence (AI) solutions that not only facilitate dialogue but also ensure that civic engagement is secure, equitable, and respectful of individual privacy rights.

Work Package 3 (**WP3 – ITHACA Platform Development**) is dedicated to the **technical design, development, and integration** of the ITHACA platform, ensuring it meets the functional and operational requirements necessary to support civic participation through AI-driven tools. WP3 encompasses a range of tasks, starting with the **collection and analysis of technical requirements (T3.1)** and the **design of the system architecture (T3.2)**. It then progresses to the **development of key platform components**, including **evidence extraction from online discussions (T3.3)**, **cognitive and social agents (T3.4)**, **ontology development (T3.5)**, **gamification mechanisms (T3.6)**, **a Public AI Register (T3.7)**, **argument visualization (T3.8)**, and **a trust and security infrastructure (T3.9)**. Additionally, it incorporates a **Personal Information Management System (T3.10)** to ensure data privacy and user control. Finally, WP3 oversees the **system integration, testing, and deployment of the platform (T3.11)**, ensuring the seamless operation of all components. By structuring the development process in a modular and scalable manner, WP3 lays the foundation for the ITHACA platform to effectively support citizen engagement, AI transparency, and trust in democratic processes.

1.2 Purpose of Deliverable D3.1

Deliverable D3.1 – ITHACA Technical Requirements, Architecture Design, and 1st Release Report serves as a foundational document outlining the early-stage technical specifications, system architecture, and initial implementation of the ITHACA platform. The purpose of this deliverable is to consolidate insights from multiple work packages (WP1, WP2, and WP5) to define the fundamental requirements necessary for the platform's development. It provides a structured analysis of the key functionalities needed to support the verification of project results, ensuring that the platform aligns with user needs, ethical considerations, and interoperability standards.

In addition to specifying the technical requirements, D3.1 also presents the conceptual architecture of the platform, detailing the core components, their interactions, and the underlying technologies. This deliverable lays the groundwork for future iterations by offering a preliminary version of the platform's structure, including data processing, security infrastructure, and integration mechanisms. Furthermore, it introduces essential elements such as evidence extraction from citizen discussions, cognitive agents, argument visualization, and gamification incentives. By documenting these aspects, D3.1 provides a roadmap for subsequent refinements, leading to the final platform release outlined in D3.2.

1.3 Structure of the Document

This document is structured to provide a comprehensive overview of the **technical requirements, system architecture, and first release** of the ITHACA platform. It is organized into distinct chapters, each focusing on a critical aspect of the platform's development, ensuring clarity and coherence in the presentation of findings.

- **Chapter 1: Introduction** – This chapter outlines the background, objectives, and purpose of Deliverable D3.1, providing context for the platform's technical development. It also describes the structure of the document.
- **Chapter 2: Technical Requirements Analysis** – Based on the outcomes of Task 3.1, this chapter defines the **technical requirements** of the ITHACA platform. It consolidates findings from WP1, WP2, and WP5 (DPIA), outlining the key functionalities necessary for platform validation.
- **Chapter 3: System Architecture Design** – This chapter presents the **conceptual architecture** of the ITHACA platform, as developed in Task 3.2. It details the main components, their interactions, selected technologies, and the overall system framework.
- **Chapter 4: Evidence Extraction from Citizens' Discussions** – Based on Task 3.3, this chapter explains how **text analytics and sentiment analysis** will be used to extract insights from multilingual online discussions, including content from news sources and social media in the pilot cities.
- **Chapter 5: Cognitive and Social Agents** – This chapter, aligned with Task 3.4, describes the development of **Natural Language Processing (NLP)-based cognitive and social agents**, including automated moderation and key phrase extraction to identify discussion trends.
- **Chapter 6: Ontology Development** – Covering Task 3.5, this chapter details the **ontology design** for structuring the ITHACA information space. It includes the integration of semantic annotation, indexing, and search functionalities.
- **Chapter 7: Gamification Incentive Mechanism** – This chapter, corresponding to Task 3.6, explains the development of **user engagement mechanisms**, including reputation-based ranking and reward systems to encourage participation in discussions.
- **Chapter 8: Public AI Register** – Based on Task 3.7, this chapter describes the design and implementation of the **Public AI Register**, which provides transparency regarding AI-driven decisions and algorithmic processes used in ITHACA.
- **Chapter 9: Argument Visualization** – This chapter, aligned with Task 3.8, presents the **argument visualization component**, which visually represents citizens' opinions on different topics using heat maps and ranked pros and cons.
- **Chapter 10: Trust and Security Infrastructure** – Addressing Task 3.9, this chapter details the **security framework** for ensuring platform integrity, data protection, and user trust, following best practices and relevant security standards.
- **Chapter 11: Personal Information Management System (PIMS)** – This chapter, corresponding to Task 3.10, describes the **PIMS functionalities**, including user-controlled data privacy settings, consent management, and compliance with data protection principles.

- **Chapter 12: System Integration, Testing, and Deployment** – Based on Task 3.11, this chapter covers the **initial integration, testing, and deployment** of the ITHACA platform, focusing on system scalability, CI/CD strategies, and overall usability.
- **Chapter 13: Conclusion** – The final chapter summarizes the findings of the deliverable, highlighting the progress made in defining the platform’s technical framework and outlining the next steps leading to the refined version in D3.2.

2. Technical Requirements Analysis (T3.1)

Task T3.1 defines the technical requirements for the ITHACA platform by consolidating insights from early-stage participatory design activities (WP1, WP2), legal and ethical assessments (WP5), and architectural feasibility studies (T3.2). The primary objective is to ensure that the technical foundations of the platform align with user needs, privacy obligations, and the broader goal of promoting democratic participation through AI-enabled tools.

The methodology followed involved:

- Stakeholder engagement with municipal authorities and civic users during WP1 pilots to collect real-world needs.
- Identification of participatory use cases requiring AI support (e.g., debate moderation, argument structuring).
- Systematic review of GDPR obligations under WP5, including a Privacy by Design evaluation and the initiation of a Data Protection Impact Assessment (DPIA).
- Technical feasibility checks by development partners based on the microservice-based system architecture and modular deployment model.

This process produced a set of functional and non-functional requirements that serve as the foundation for design and implementation across all technical tasks in WP3.

2.2 Functional Requirements

The ITHACA platform must provide a set of coherent, user-centric functionalities that allow citizens to engage in civic discourse, submit ideas, debate proposals, and interact with local governance structures. The major functional areas are as follows:

2.2.1 User Registration, Authentication, and Roles

- Integration with Keycloak (open-source Identity and Access Management) to support OAuth2 and OpenID Connect for secure authentication and role management.
- Support for multiple registration methods (email/password, social login) with strict adherence to the principle of data minimization: only essential personal data (e.g., email, city) are collected.
- Differentiated user roles: citizen, municipal official, moderator, admin – each with clearly defined access rights.

2.2.2 Civic Forum and Topic Discussions

- Citizens must be able to create public topics or proposals, contribute comments, and vote on content.
- Features include:
 - Post creation with text, file attachment, and optional anonymity.
 - Accessibility features: speech-to-text input, language translation, simplification, and summarization.
 - Post interactions: upvote/downvote, report inappropriate content, mark helpful, bookmark.
- Content filtering and navigation by topic, category, popularity, and recency.

2.2.3 Argument Visualization and Deliberation Tools

- A visual interface for browsing public issues, user-generated proposals, and related arguments (supporting or opposing).
- Dynamic visualizations (e.g., pie charts) display vote-weighted popularity of proposals to support deliberative engagement without biasing visibility.
- Argument submission interfaces include optional tagging of pros/cons and structured debate elements.

2.2.4 AI Tools and Transparency Mechanisms

- **Public AI Register:** A centralized, user-friendly listing of AI components used in the platform (e.g., toxicity detection, content clustering), including descriptions of logic, training data, and limitations.
- **Toxic Speech Detection (TSD):** An NLP-based module to identify and flag offensive content. Operates in a “human-in-the-loop” configuration where moderators receive AI suggestions but retain final authority.
- **AI Fairness & Privacy Dashboards:** Components to display insights about AI performance and ensure transparency in automated decision-making.

2.2.5 Gamification and Motivation Mechanisms

- A points and badge system based on user engagement levels:
 - **Activity badge** (e.g., login frequency, votes cast),
 - **Competence badge** (e.g., proposals submitted, comments written),
 - **Relatedness badge** (e.g., collaborative actions, helping others).
- Leaderboards are optional and anonymizable to preserve user agency and avoid negative social pressure.
- Clear “how to earn” explanations ensure transparency and foster intrinsic motivation (per Self-Determination Theory).

2.2.6 Personal Information Management System (PIMS)

- Allows users to:
 - View and edit personal data,

- Manage cookie and data consent,
- Request data download (data portability),
- Delete accounts (right to erasure).
- Consent is granular (e.g., per non-essential cookie or AI tool) and revocable.

2.2.7 Moderation and Complaint Handling

- Reporting flow: posts and comments can be flagged for predefined reasons (e.g., hate speech, misinformation).
- Moderation transparency: Users are notified of outcomes (e.g., content removal, warnings).
- Complaint mechanism: A formal process for users to contest moderation decisions.

2.3 Non-Functional Requirements

2.3.1 Legal and Ethical Compliance

- Full alignment with GDPR principles: lawfulness, fairness, transparency, purpose limitation, data minimization, accuracy, storage limitation, and accountability.
- DPIA-informed design choices, such as no pre-checked consent boxes, default denial of non-essential cookies, and prohibition of automated decisions without human oversight.
- Specific handling of special cases:
 - Anonymous posts: clarified risks, logged IP (retained by platform), and legal disclaimers.
 - Toxicity detection: includes a non-AI fallback to ensure lawful processing in sensitive cases.

2.3.2 Architecture, Integration, and Deployment

- **Microservices architecture**, with containerization via Docker and orchestration via Docker Compose.
- Backend services communicate over secure HTTP and WebSockets using internal and external API layers, protected by reverse proxy (NGINX) and SSL certificates.
- **CI/CD pipelines** enable rapid integration and testing across modules.
- Platform divided into modular dashboards for core functionalities (forum, arguments, chat, gamification) and verticals (AI fairness, cybersecurity, privacy).

2.3.3 Performance and Scalability

- The platform must scale to support simultaneous civic engagement activities in multiple cities.
- Support for horizontal scaling of critical services (e.g., NLP pipeline, TSD engine).
- Logging, monitoring, and analytics (e.g., Matomo, Google Analytics) configured in a GDPR-compliant way to assess performance and participation trends.

2.3.4 Usability and Accessibility

- Compliance with WCAG 2.1 Level AA.
- Mobile-responsive interface, keyboard navigation, screen-reader support.
- Simplification tools for low-literacy and non-native users.

- Help center, searchable FAQ, and onboarding content (to be implemented).

The technical requirements for the ITHACA platform articulate the project's commitment to building a democratic, inclusive, and ethically sound digital environment. These requirements balance usability with legal rigor and provide a blueprint for development across all WP3 tasks. As the platform evolves through testing and refinement, these specifications will remain a central reference to ensure the system continues to meet the expectations of users, regulators, and society at large.

3. System Architecture Design (T3.2)

3.1 Introduction

The architectural design of the ITHACA platform is foundational to its vision of enabling ethical, inclusive, and scalable civic engagement through artificial intelligence. Task T3.2 addresses this goal by defining the conceptual and technical architecture required to integrate diverse platform components—ranging from AI agents and multilingual data pipelines to semantic reasoning tools and citizen-facing dashboards—into a cohesive, modular system.

Building on the functional and non-functional requirements identified in Task T3.1, the architecture outlined in this chapter adopts a **hybrid model**, combining microservices and event-driven messaging. This approach provides the necessary flexibility, scalability, and fault tolerance to support a distributed, multilingual civic engagement platform operating across different cities and user groups.

The system architecture reflects ITHACA's commitment to:

- **Human-centric AI:** Embedding fairness, transparency, and explainability into system design.
- **Privacy and security by design:** Ensuring GDPR compliance, secure data flows, and user-controlled information management.
- **Adaptability:** Supporting continuous deployment, modular feature upgrades, and integration with evolving components from other work packages (e.g., cognitive agents, gamification, PIMS).
- **Interoperability:** Enabling seamless communication between internal services and external systems through open standards and API-driven design.

This chapter presents an in-depth look at the architectural principles guiding ITHACA's development. It introduces the rationale behind choosing a hybrid system model, explores key patterns and technologies—including microservices, containerization, orchestration, and asynchronous event messaging—and describes how these choices translate into a robust, layered platform that can evolve in step with user needs, regulatory changes, and technical innovation.

3.1.1 Purpose of the Platform

Scope:

The scope of the ITHACA project is to delve into the complexities and controversies surrounding the integration of Artificial Intelligence (AI) in the domain of civic participation. The project aims to advance

the understanding of these complexities and stimulate responsible practices in research and innovation. It specifically seeks to anchor the design of AI technologies in human rights, values, ethical principles, and moral reflections. The focus is on developing an AI-driven online discussion platform tailored for civic participation in local governance.

Objectives:

1. **Maximize Positive Impact:** The primary objective is to contribute to the positive impact of AI on democratic institutions and processes. This involves developing and testing a human-centric AI platform that prioritizes ethical considerations and values.
2. **Human-Centric AI Design:** The project envisions the creation of an AI platform that incorporates human and social interpretations into the design process. This approach ensures that AI decision-making processes are explainable, transparent, and fair.
3. **Focus on Democratic Processes:** ITHACA aims to enhance the functioning of democratic institutions by fostering civic participation in local governance through the AI platform. The emphasis is on developing a tool that empowers citizens and provides a mechanism for their voices to be heard in decision-making processes.
4. **Addressing Vulnerable Groups:** Special attention is given to the needs and voices of citizens from vulnerable groups. The project aims to comprehensively monitor, debate, and address human rights impacts resulting from the design, development, and deployment of algorithmic systems.
5. **Pilot Activities in EU Countries:** The project plans to conduct pilot activities in two different European Union countries. These activities will assess the effectiveness of AI in enhancing citizens' participation in local governance while minimizing adverse effects.
6. **Preventing Adverse Effects:** ITHACA seeks to prevent or minimize adverse effects arising from the unregulated use of AI. This includes empowering citizens with knowledge about the AI models involved in decision-making processes.
7. **Scalable Tools:** The project aims to provide scalable tools for identifying security risks and threats. Additionally, it seeks to ensure conformity with fairness and privacy principles in AI-based systems and big data for civic engagement applications.
8. **Data Governance Framework:** Based on implementation experiences, ITHACA aims to define a data governance framework that incorporates relevant ethical principles and practices. This framework will guide the design, development, and deployment processes for AI civic engagement platforms.
9. **Policy Recommendations:** ITHACA intends to provide policy recommendations on philosophical, legal, and ethical values that should be embedded in the development of AI civic engagement platforms. This includes guidance on ensuring responsible and ethical AI practices in civic participation.

3.1.2 Target audience

The target audience for the EU-funded project ITHACA includes various stakeholders involved in civic participation, local governance, AI development, and policy-making within the European Union. The project aims to engage with a diverse set of actors to achieve its objectives. The key target audience for ITHACA may include:

1. **Citizens and Civil Society Organizations:**

- Engaging citizens to ensure their active participation in local governance and decision-making processes.
- Collaborating with civil society organizations that advocate for human rights, transparency, and ethical considerations in AI applications.

2. **Local Governance Authorities:**

- Working with local government authorities and officials responsible for civic engagement and governance.
- Providing tools and insights to enhance citizen participation in local decision-making.

3. **AI Developers and Researchers:**

- Collaborating with AI developers and researchers to ensure responsible and ethical AI design.
- Providing insights and tools for incorporating human-centric approaches and ethical considerations into AI development.

4. **Policy-Makers and Legislators:**

- Engaging with policymakers and legislators to influence the development of regulations and policies related to AI in civic participation.
- Offering policy recommendations based on project findings.

5. **Academic and Research Community:**

- Sharing project findings with the academic and research community to contribute to the collective knowledge on the intersection of AI, civic participation, and ethics.
- Encouraging further research in this domain.

6. **Ethics and Privacy Advocates:**

- Collaborating with individuals and organizations advocating for ethics and privacy in AI applications.
- Ensuring that the project addresses concerns related to fairness, transparency, and privacy in civic engagement platforms.

7. **EU Institutions:**

- Engaging with relevant European Union institutions involved in shaping policies related to AI, civic engagement, and human rights.
- Contributing to the broader EU strategy on responsible AI development.

8. **Vulnerable and Underrepresented Groups:**

- Paying particular attention to the needs and voices of citizens from vulnerable and underrepresented groups.
- Ensuring that the project's outcomes address potential biases and challenges faced by these groups in AI-driven civic participation.

9. **General Public:**

- Raising awareness among the general public about the project's goals, findings, and the importance of responsible AI in civic engagement.
- Empowering citizens with knowledge about AI models and their involvement in local governance.

By targeting a diverse set of stakeholders, ITHACA aims to create a comprehensive and inclusive approach to the development and implementation of AI technologies in civic participation within the European Union.

3.2 Overview of Hybrid Architectures

3.2.1 Definition of Hybrid Architectures

Hybrid Architectures represent a strategic blend of distinct architectural paradigms, combining the strengths of different approaches to create a unified and flexible system. In the context of this document, the hybrid architecture specifically integrates principles from both Event-Driven Messaging and Microservices architectures.

Characteristics of Hybrid Architectures:

- **Modularity:** Hybrid architectures leverage modular design principles, allowing components to be loosely coupled and independently deployable.
- **Scalability:** The combination of Microservices and Event-Driven Messaging enables scalable solutions that can adapt to varying workloads.

Integration of Event-Driven and Microservices Concepts:

- **Event-Driven Communication:** Hybrid architectures promote communication through events, allowing microservices to react to changes in the system in real-time.
- **Microservices Independence:** Each microservice within the hybrid architecture retains its autonomy, while event-driven communication facilitates coordination and collaboration.

3.2.2 Advantages and Challenges

Advantages of Hybrid Architectures:

- **Flexibility and Adaptability:** Hybrid architectures offer flexibility by allowing the incorporation of different technologies, making it easier to adapt to evolving business requirements.
- **Scalability:** The modular nature of Microservices combined with event-driven communication enhances scalability, enabling efficient handling of increased workloads.

Challenges of Hybrid Architectures:

- **Complexity:** Integrating Microservices and Event-Driven Messaging introduces complexity in terms of design, development, and maintenance.
- **Consistency and Coordination:** Ensuring consistency across microservices and managing coordination through events may pose challenges, especially in distributed systems.

3.3 Microservices Architecture

Microservices Architecture is a contemporary software development approach that structures an application as a collection of independently deployable and loosely coupled services. Each service, or microservice, is designed to perform a specific business function and communicate with others through well-defined APIs. This architectural style contrasts with monolithic applications, offering benefits such as flexibility, scalability, and faster development cycles.

Key characteristics of Microservices include:

3.3.1 Service Independence

Microservices emphasize the independence of each service, allowing them to be developed, deployed, and scaled independently. This characteristic fosters agility and enables teams to work on specific services without affecting the entire application. Service independence also facilitates the use of different programming languages and technologies for each microservice.

3.3.2. Scalability

Scalability is a core aspect of Microservices Architecture. Each microservice can be scaled independently based on its specific workload, responding dynamically to changing demand. This granular scalability enhances resource utilization and ensures optimal performance, even in scenarios with varying levels of user activity.

3.3.3 Maintainability

Maintainability in Microservices Architecture is improved due to the modular nature of services. Developers can focus on individual microservices, making it easier to understand, update, and troubleshoot code. This modularity simplifies maintenance tasks, reduces the risk of introducing errors, and allows for faster and more frequent releases.

Design Principles

3.3.4 Decentralized Data Management

Microservices often embrace decentralized data management, with each service responsible for its own database. This decentralization minimizes dependencies between services and reduces the likelihood of a single point of failure. However, it requires careful consideration of data consistency and synchronization between services.

3.3.5 API-Based Communication

API-based communication is fundamental to Microservices Architecture. Services communicate through well-defined APIs, often using lightweight protocols such as REST or messaging queues. This design principle ensures loose coupling between services, allowing for independent evolution and versioning.

Implementation Best Practices

3.3.6 Service Discovery

Service discovery is crucial for managing the dynamic nature of Microservices Architecture. Tools and frameworks for service discovery help services locate and communicate with each other in a distributed environment. This capability becomes essential as the number of microservices grows.

3.3.7 Containerization

Containerization, commonly using technologies like Docker, facilitates packaging microservices and their dependencies into isolated containers. Containers ensure consistency across different environments, simplify deployment, and enhance scalability. Container orchestration platforms, like Kubernetes, further streamline container management.

3.3.8 Orchestration

Orchestration tools, particularly Kubernetes, automate the deployment, scaling, and management of containerized applications. Kubernetes simplifies the complexities of managing microservices, provides resilience, and ensures high availability. It also supports features like rolling updates and load balancing.

3.3.9 Microservices Security Considerations

Microservices introduce specific security challenges that require careful attention:

- **Network Security:** As microservices communicate over networks, securing communication channels through encryption and authentication is crucial to prevent unauthorized access and data breaches.
- **Identity and Access Management:** Implement robust identity and access management mechanisms to control service access. This involves enforcing authentication and authorization policies to ensure that only authorized entities interact with microservices.

- **Data Security:** Since each microservice may have its own database, data security becomes paramount. Implement encryption for sensitive data, enforce proper access controls, and regularly audit data storage practices.
- **Monitoring and Logging:** Incorporate comprehensive monitoring and logging to detect and respond to security incidents promptly. Monitor service interactions, system logs, and user activities to identify potential vulnerabilities.
- **Securing APIs:** Ensure that APIs used for communication between microservices are secure. Implement measures such as rate limiting, input validation, and API key management to prevent malicious attacks and data manipulation.
- **Container Security:** When using containerization, adopt container security best practices. Regularly update container images, scan for vulnerabilities, and employ tools to monitor and manage container security.

3.4 Event-Driven Messaging Architecture

Event-Driven Architecture (EDA) is a paradigm that focuses on the production, detection, and consumption of events. In this architectural style, components within a system communicate through events, representing significant occurrences or changes in state. Event-Driven Messaging Architecture leverages this approach to facilitate loosely coupled, asynchronous communication between different parts of a system.

The key concepts of Event-Driven Architecture are:

Events

Events are fundamental to Event-Driven Architecture. They represent occurrences or state changes within the system, conveying information about what has happened. Events are typically structured pieces of data that contain relevant details, and they serve as triggers for other components to react.

Event Producers and Consumers

- **Event Producers:** Components or services that generate and emit events. These producers are responsible for signaling changes in the system, making events available for consumption.
- **Event Consumers:** Components or services that subscribe to and react to events. Consumers are designed to respond to specific types of events and perform actions accordingly.

3.4.1 Event-Driven Design Patterns

Publish-Subscribe

The Publish-Subscribe pattern is a fundamental design pattern in Event-Driven Architecture. In this pattern, event producers (publishers) distribute events to a group of event consumers (subscribers) without their direct knowledge. This decoupling enables a scalable and flexible communication model, where multiple components can react to the same event independently.

Event Sourcing

Event Sourcing is a design pattern that involves capturing all changes to an application state as a sequence of events. Instead of storing the current state of an entity, the system maintains a history of events that led to the current state. This pattern is particularly valuable for auditing, debugging, and reconstructing system states.

.4.2 Implementation Technologies

Message Brokers

Kafka: Kafka is a distributed streaming platform that excels in handling large-scale, fault-tolerant event streams. It provides durability, high throughput, and the ability to handle real-time data processing.

Event Hubs

Event Hubs are cloud-based, scalable event streaming services. Offered by various cloud providers, such as Azure Event Hubs, they provide a fully managed platform for ingesting, processing, and analyzing massive amounts of event data.

3.4.3 Event-Driven Microservices Communication

Event-Driven Messaging Architecture is closely aligned with Microservices Architecture, enhancing communication and collaboration between microservices:

- **Loose Coupling:** By relying on events, microservices become loosely coupled, allowing them to operate independently and evolve without direct dependencies on each other.
- **Scalability:** Event-Driven Microservices Communication supports scalable and resilient systems. Microservices can react to events in real-time, making it possible to adapt to changing conditions and distribute workloads efficiently.
- **Flexibility:** Event-Driven communication enables flexibility in system design. New microservices can be introduced or existing ones modified without disrupting the entire system, promoting agility and adaptability.

3.5 Integration of Microservices and Event-Driven Messaging

3.5.1 Rationale for Integration

The integration of Microservices and Event-Driven Messaging is driven by the desire to create a system that combines the benefits of both architectural paradigms. The rationale includes achieving enhanced flexibility, scalability, and responsiveness in modern software development. By integrating Microservices with Event-Driven Messaging, organizations can build systems that are capable of adapting to changing conditions, handling real-time events, and supporting a distributed and modular architecture.

Challenges and Solutions

3.5.2 Consistency and Transactionality

Challenges:

- Ensuring consistency across microservices when processing events asynchronously.
- Maintaining transactional integrity across distributed components.

Solutions:

- **Eventual Consistency:** Accepting that consistency may not be immediate but will eventually be achieved. Systems implement mechanisms to reconcile eventual inconsistencies.
- **Transactional Outbox Pattern:** Using the transactional outbox pattern to ensure that events are produced within the same transaction as the state changes, maintaining consistency.

3.5.3 Event Schema Evolution

Challenges:

- Evolving event schemas over time without disrupting the entire system.
- Ensuring backward and forward compatibility as events evolve.

Solutions:

- **Schema Versioning:** Implementing versioning for event schemas to handle changes gracefully.
- **Schema Registry:** Using a schema registry to manage and enforce compatibility rules for evolving events.

3.6 Design Patterns for Hybrid Architectures

3.6.1 Command Query Responsibility Segregation (CQRS)

CQRS separates the read and write responsibilities of a system. It introduces different models for reading and writing data. Write operations (commands) update the system state and produce events, while read operations (queries) are optimized for data retrieval.

Benefits include improved scalability (read and write operations can scale independently) and flexibility (enabling the use of different data storage and retrieval mechanisms for reads and writes).

An issue to consider is increased complexity: introducing separate models for reading and writing may add complexity to the system.

3.6.2 Saga Pattern

The Saga Pattern manages long-lived transactions in a distributed environment by breaking them into a sequence of smaller, more manageable steps. Each step in the saga corresponds to a transactional operation and is associated with an event.

Among the benefits of Saga Pattern, we can observe distributed transaction management, which allows for coordination of actions across multiple microservices, and fault tolerance: if a step in the saga fails, compensating actions can be triggered to maintain system integrity.

On the other hand, sagas may operate under eventual consistency, and careful design is required to handle failures and compensating transactions.

3.7 End-to-End Architecture Definition

3.7.1 Introduction

This section provides an overview of the end-to-end system architecture, describing how various components and layers interact to provide a cohesive application environment.

3.7.2 Layers of the application

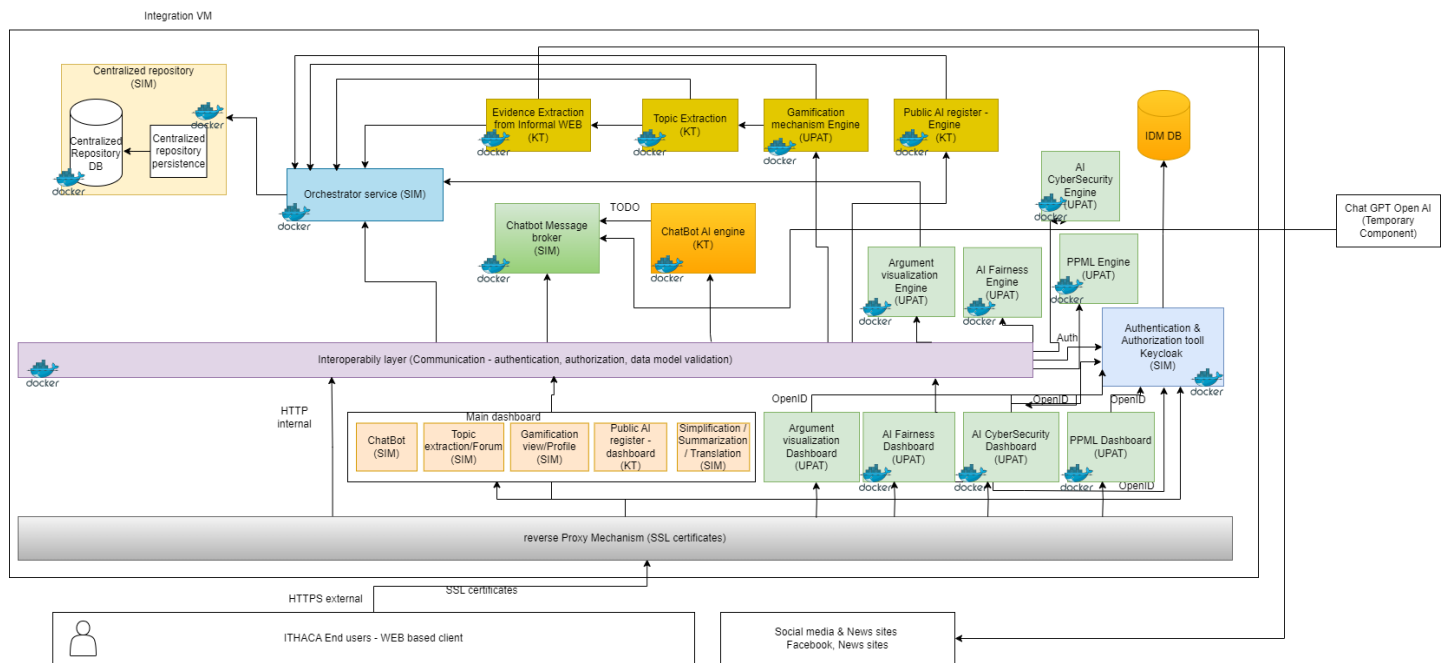


Figure 1 Layers of the application

Platform internal components:

- Centralized repository
 - Structured DataBase for storing all the common data generated and consumed by the platform components
 - Dedicated persistence service which stores and interrogates data in and from the DB
- Business layers
 - All the processing services, AI services
 - Broker service for chat
- Interoperability layer

- Communication service, dedicated for handle external communication
- Orchestrator services, dedicated for internal and private communications between internal cloud components
- Identity Manager & authentication layer
 - Keycloak Authentication mechanism
 - Keycloak DataBase repository
- Dashboard layers
 - Common platform dashboard
 - Argument visualization dashboard
 - AI Fairness dashboard
 - AI Cybersecurity dashboard
 - PPML dashboard

Platform external tools:

- End users
- Social media app – Facebook
- News web sites

This multi-layered architecture ensures modularity, scalability, and security while providing a clear separation of concerns.

3.7.3 Semantic Layer architecture

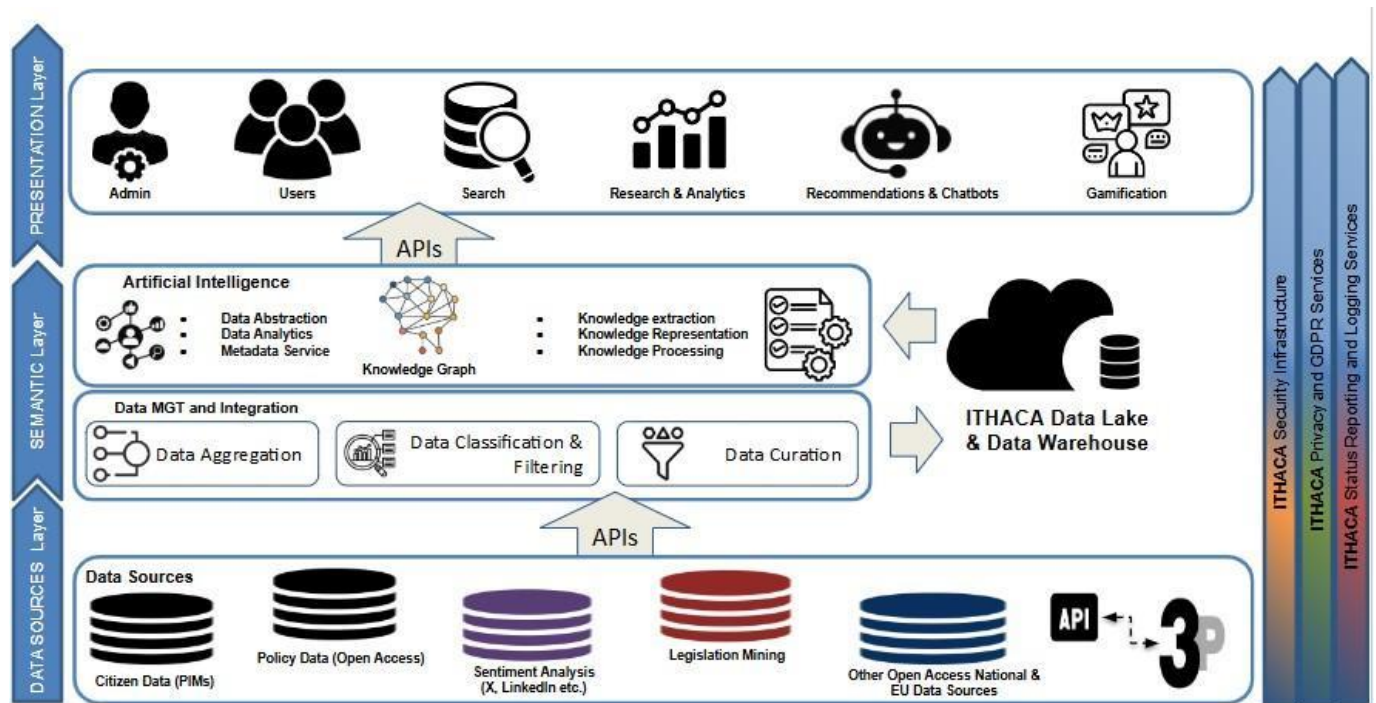


Figure 2 Semantic Layer architecture

3.7.4 Summary of Tools used in All Use Cases

Table 1 Summary of Tools used in All Use Cases

Code	Description
T1 - Evidence extraction from citizen's discussions on the informal web	Collecting multilingual (Romanian and Slovak) content from the Web, including news sources from the two pilot cities and public social media content (e.g., social media pages of the pilot partners) to gain a deeper understanding of citizens' opinions with sentiment analysis by identifying how citizens feel about the different subjects of the discussion.
T2 - Topic Extraction	NLP tools to cluster text posts by citizens on the ITHACA platform and the informal web (content search, selection, acquisition, categorization). Furthermore, key phrase extraction techniques will be used to understand common topics and trends, such as the discussion topics.
T3 - Toxicity Detection Tool / Automated moderation	An automated moderator "bot" to assist with human-performed moderation, enabling the identification of inappropriate content submitted by users.
T4 - Ontology	Systematic organizing of the ontological knowledge and design of the ITHACA information space reusing/developing ontologies in the following fields: e-government public services. This task will carry out the integration of the ontology building and maintenance, semantic annotation, semantic indexing, search and retrieval, and semantic reasoner.
T5 - Gamification incentive mechanism	An incentive mechanism (e.g., reputation system) will be developed by T3.6 that will cluster the users of ITHACA based on their participation measure into different classes and will award points toward one's reputation based on his/ her contribution to the discussions hosted by the ITHACA platform.

T6 - Public AI Register	Design and implementation of the webpages of the Public AI registers (in the pilot cities languages) and development of the content of the registers, providing information about the decisions and assumptions that were made in the process of developing, implementing, managing and ultimately dismantling algorithms used in ITHACA platform
T7- Argument Visualization	Development of the Argument Visualization component which will present a visual representation of citizens' opinions concerning different ideas proposed by other citizens, along a scale of agree to disagree, together with a list of ranked pros and cons. This component will visualize the citizens' different ideas and the degree of acceptance these ideas gain by other citizens using a heat map. In this way, the most accepted ideas for each subject will be apparent to everyone.
T8 - AI Fairness	An AI Fairness tool whose purpose is to conform with the fairness principle, to promote equality, inclusivity and oppose discrimination as well as render the evaluated AI system more trustworthy.
T9 - PPML Tool	Privacy Preserving Machine Learning Tool to employ data concealment techniques to safeguard user privacy.
T10 - AI CyberSecurity Tool	An AI Cybersecurity tool to detect possible security breaches and threats within the AI models. An open-source tool named ModelScan that scans for malicious code that may impose security vulnerabilities in AI systems is proposed as the AI Cybersecurity tool.
T11-Simplification Tool	A tool that simplifies a very complex text
T12-Summarization Tool	A tool that summarizes a long text
T13 – Translation Tool	A tool that will translate a text to the selected language. Translation capabilities are from Romanian and Slovak to English and from English to Romanian and Slovak.
T14 - Chatbot	A tool to assist platform navigation and provide information to users, enhancing user experience

3.7.5 Component Interaction and Data Flow

This section describes how data flows through the system, starting from the client request to the backend processing and database interactions, and back to the client. The API authorization will be granted via a bearer token. The main “bucket” of the database is a collection. We can have multiple collections of artefacts, as we need.

The API endpoints for it are the following:

GET localhost:8090/api/collections

Retrieves the list of collections, see example below

Possible types of collections: General (no type/category of interest specified), Transport, Budget, etc. (further investigation pending)

Possible status for the collections are:

PENDING, IN_PROGRESS, COMPLETED, FAILED, RETRYING, PAUSED, CANCELED.

```
[
  {
    "id": "6673fc9cdb03cc0b2902dde5",
    "place": "Martin",
    "type": "General",
    "status": "PENDING",
    "whitelist": [ // links to be crawled
      "https://random-news-link.com",
      "https://random-news-link1.com"
    ],
    "blacklist": [ //can remain blank - list of links that have to be ignored in crawling
      "https://random-news-link-to-ignore.com",
      "https://random-news-link1-to-ignore.com"
    ],
    "keywords": [
      "tag1","tag2" //can remain blank
    ]
  },
  {
    "tags": [
      "tag1",
      "tag2"
    ],
    "name": "Martin1",
    "created": "2024-06-20T12:55:40.708Z",
    "updated": "2024-06-21T12:56:12.122Z",
    "description": "Martin1 description"
  },
  {
    "id": "6673fcbcdb03cc0b2902dde6",
    "type": "General",
    "place": "Brasov",

    "status": "PENDING",
    "whitelist": [

      "https://random-news-link.com",
      "https://random-news-link1.com"
    ],
    "blacklist": [ //can remain blank
      "https://random-news-link-to-ignore.com",
      "https://random-news-link1-to-ignore.com"
    ]
  }
]
```

```

    ],
    "keywords": [
        "tag1", "tag2" //can remain blank
    ],
    "tags": [
        "tag1",
        "tag2"
    ],
    "name": "Brasov1",
    "created": "2024-06-20T12:56:12.122Z",
    "updated": "2024-06-21T12:56:12.122Z",
    "description": "Brasov1 description"
}
]

```

POST localhost:8090/api/collections

Adds a new collection

```

{
    "type": "General",
    "status": "PENDING",
    "place": "Brasov",
    "whitelist": [
        "https://random-news-link.com",
        "https://random-news-link1.com"
    ],
    "blacklist": [
        //can remain blank
        "https://random-news-link-to-ignore.com",
        "https://random-news-link1-to-ignore.com"
    ],
    "keywords": [
        "tag1", "tag2" //can remain blank
    ],
    "tags": ["tag1", "tag2"],
    "name": "Brasov1",
    "description": "Brasov1 description"
}

```

PUT localhost:8090/api/collections/{collectionId}

Updates a collection. The body will be a collection

DELETE localhost:8090/api/collections/{collectionId}

Deletes a collection

Every collection can have multiple artefacts, the result of the web scraping. The fields of the artefact are the following:

```

@Id private String id;
@NotBlank private String domainId;
@NotBlank private String type;
private String created;
private List<String> tags;
private Map<String, List<String>> attributes;
private String published;
private String content;
private String name;
private String collectionId;

```

GET localhost:8090/api/collections/{collectionId}/artefacts

Retrieves an array of the artefacts. It also supports filtering by a certain tag, type, domainId, page, pageSize, order.

GET localhost:8090/api/collections/{collectionID}/artefacts/{domainId}

Retrieves an array with the artefact that has that domainId, should be unique

POST localhost:8090/api/collections/{collectionId}/artefacts

Posts a new artefact

```
{
  "domainId": "654e15146958fe68ae353016",
  "type": "General",
  "created": "2023-11-10T13:33:40.863Z",
  "tags": [
    "tag1"
  ],
  "attributes": {
    "link": [
      "https://random-news-link.com"
    ],
    "attribute2": [
      "text"
    ]
  },
  "published": null,
  "content": "Some random text found on a news page",
  "name": "Title",
  "collectionId": "654a44af1db843352b9d409d"
}
```

There is also a topics entity that has the following fields, where the topics of discussions extracted from the crawled content will be stored:

```
@Id private String id;
@NotBlank private String domainId;
@NotBlank private String type;
@NotBlank private String place; // "Brasov" or "Martin"
private String created; // date string of creation of the format
private List<String> tags;
private Map<String,List<String>> attributes;
private String published;
private String content;
private String title;
private String name;
private List<String> likes; // an array with all the users that liked
the topic
private List<String> dislikes; // an array with all the users that
disliked the topic
private int views;
```

The flow of the components will be the following:

When the **T1-Evidence extraction from citizen's discussions on the informal web** goes online, it will look for the collections with the status `IN_PROGRESS`

```
GET localhost:8090/api/collections?status=IN_PROGRESS
```

and it will finish crawling them one by one, then when the crawling is done, it will modify the status of the collection to `COMPLETED` if successful or `FAILED` if an error occurred. It will also update the updated field with current date/time.

Then it will look for collections with status `PENDING`, these are the newly created collections and will crawl them. When finishing it will modify the status of the collection to `COMPLETED` if successful or `FAILED` if an error occurred.

Then it will look for collections with status `FAILED`. When finishing it will modify the status of the collection to `COMPLETED` if successful or `FAILED` if an error occurred. (avoiding infinite loops is to be discussed).

The second component, **T2-TopicExtraction**, will start at a fixed hour every day, retrieve the list of the existing topics (`GET localhost:8090/API/topics`) and check if any of them were updated since the last time.

It will then check if there are any new topics of discussion besides the already discovered ones and `POST` them (`POST localhost:8090/API/topics`).

T14 – Chatbot – this component will listen to a Kafka topic for queries (alternatives are being researched)

T101_CHAT_REQUEST

The format of the Message is:

```
{
  "id": "667541729da6733bc368b0f5",
  "from": "actionadi1",
  "to": "chatbot",
  "content": "Who are you?",
  "sent": "2024-06-21T12:01:38.482Z",
  "type": "user"
},
```

It will send a response when the analysis is done to the following topic (with the possibility of calling a backend API):

T102_CHAT_RESPONSE

The format of the Message is:

```
{
  "id": "667541749da6733bc368b0f6",
  "from": "chatbot",
```

```

    "to": "actionadil",
    "content": "I am a language model AI created by OpenAI. I am here to assist you with
any questions or tasks you may have. How can I help you today?",
    "sent": "2024-06-21T12:01:40.660Z",
    "type": "client"
}

```

For the T5-Gamification **incentive mechanism** an up vote / down vote for every topic on every user post is being designed, so it will be possible to have access to any user's up-down votes, number of posts through an API.

Communication with the **Toxicity Detection Tool** will be done in the following format:

When a user posts a new post on a certain topic, an API call will be done to the Toxicity Detection Tool to analyze:

```

{
  "type": "post",
  "content": "Random text to be analyzed by the toxicity tool",
  "topic": "667541749da6733bc368b0f6", //id of the topic
  "post": "6675427b9da6733bc368b0f9" // id of the post
}

```

The response from the Toxicity Detection Tool will be sent as a response, a value of over 0.8 (threshold to be reviewed and refined depending on results) of any of the attributes will reject the post:

```

{
  "original_text": "A redacted very offensive post",
  "translated_text": "A redacted very offensive post",
  "attributes": {
    "SEXUALLY_EXPLICIT": {
      "spanScores": [
        {
          "begin": 0,
          "end": 60,
          "score": {
            "value": 0.17417414,
            "type": "PROBABILITY"
          }
        }
      ]
    },
    "summaryScore": {
      "value": 0.17417414,
      "type": "PROBABILITY"
    }
  },
  "TOXICITY": {
    "spanScores": [
      {
        "begin": 0,
        "end": 60,
        "score": {
          "value": 0.96069145,
          "type": "PROBABILITY"
        }
      }
    ]
  }
}

```

```

    }
  ],
  "summaryScore": {
    "value": 0.96069145,
    "type": "PROBABILITY"
  }
},
"THREAT": {
  "spanScores": [
    {
      "begin": 0,
      "end": 60,
      "score": {
        "value": 0.86058503,
        "type": "PROBABILITY"
      }
    }
  ],
  "summaryScore": {
    "value": 0.86058503,
    "type": "PROBABILITY"
  }
},
"PROFANITY": {
  "spanScores": [
    {
      "begin": 0,
      "end": 60,
      "score": {
        "value": 0.6919765,
        "type": "PROBABILITY"
      }
    }
  ],
  "summaryScore": {
    "value": 0.6919765,
    "type": "PROBABILITY"
  }
},
"FLIRTATION": {
  "spanScores": [
    {
      "begin": 0,
      "end": 60,
      "score": {
        "value": 0.42156047,
        "type": "PROBABILITY"
      }
    }
  ],
  "summaryScore": {
    "value": 0.42156047,
    "type": "PROBABILITY"
  }
},
"SEVERE_TOXICITY": {
  "spanScores": [
    {

```



```
"content": "A very complex text that is not easy to understand and needs to be simplified"
}
```

The simplified content will be sent to the response:

```
{
  "simplified_content": "Simplified text"
}
```

3.7.6 Implementation Guidelines

Steps for Implementing a Hybrid Architecture

Step 1: Assess Current State and Requirements

- Evaluate the existing architecture and identify areas where a hybrid approach is beneficial.
- Define the requirements, considering factors like scalability, flexibility, and real-time responsiveness.

Step 2: Define Microservices and Event-Driven Components

- Identify the services that can be decomposed into microservices.
- Determine which components can benefit from event-driven communication.

Step 3: Choose Appropriate Technologies

- Select suitable technologies for microservices and event-driven communication, considering factors such as programming languages, frameworks, and messaging platforms.

Step 4: Design Communication Protocols

- Define communication protocols between microservices and establish event-driven patterns, including publish-subscribe or event sourcing.

Step 5: Implement Microservices and Event Producers/Consumers

- Develop microservices and integrate event producers and consumers into the architecture.
- Ensure proper API design for communication between microservices.

Step 6: Implement Scalability Measures

- Incorporate mechanisms for scaling microservices independently, considering load balancing, auto-scaling, and containerization.

Step 7: Address Data Consistency

- Implement strategies for maintaining data consistency across microservices and events.
- Consider techniques such as eventual consistency, compensating transactions, and transactional outbox patterns.

Step 8: Test and Validate

- Conduct thorough testing, including unit testing, integration testing, and performance testing.
- Validate that the hybrid architecture meets the defined requirements.

Step 9: Deployment and Monitoring

- Deploy the hybrid architecture incrementally, starting with non-critical components.
- Monitor the system during and after deployment to identify potential issues.

Step 10: Iterate and Optimize

- Gather feedback from users and stakeholders.
- Iterate on the architecture based on lessons learned and identified areas for improvement.

Monitoring and Debugging Strategies

Strategy 1: Comprehensive Logging

- Implement detailed logging across microservices and event-driven components.
- Use centralized logging tools to aggregate and analyze logs for troubleshooting.

Strategy 2: Distributed Tracing

- Implement distributed tracing mechanisms to trace requests and events across microservices.
- Utilize tools like Zipkin or Jaeger to analyze traces and identify bottlenecks or issues.

Strategy 3: Real-Time Monitoring

- Implement real-time monitoring solutions to track system performance, resource utilization, and key metrics.
- Utilize tools like Prometheus or Grafana for real-time visualization of metrics.

Strategy 4: Health Checks and Alerts

- Implement health checks for microservices to detect issues proactively.
- Set up alerts based on predefined thresholds to receive notifications for potential problems.

Strategy 5: Debugging Tools

- Integrate debugging tools into the development and deployment process.
- Use tools like VisualVM, gdb, or integrated development environment (IDE) debuggers for pinpointing and resolving issues.

Strategy 6: Incident Response Plan

- Develop an incident response plan outlining procedures for addressing and resolving critical issues.
- Include steps for communication, escalation, and post-incident analysis.

Strategy 7: Continuous Monitoring and Improvement

- Continuously monitor system performance and user feedback.
- Regularly review and improve monitoring and debugging strategies based on evolving requirements and technology advancements.

By following these implementation guidelines, organizations can effectively deploy and manage hybrid architectures that leverage both Microservices and Event-Driven Messaging. The monitoring and debugging strategies outlined ensure that the system remains resilient, scalable, and responsive, with mechanisms in place to identify and address issues promptly.

3.7.7 Conclusion

In conclusion, the exploration of hybrid architectures combining Microservices and Event-Driven Messaging reveals a powerful and flexible approach to designing modern software systems. A strategic approach, a commitment to collaboration, and a willingness to adapt are required. By following best practices, staying informed about emerging technologies, and prioritizing scalability and security, organizations can build resilient, future-proof systems that meet the demands of a dynamic digital landscape.

4. Evidence Extraction from Citizen Discussions (T3.3)

4.1. Overview

This task focuses on the design of an automated pipeline for the collection and analysis of multilingual content (Romanian and Slovak) sourced from the web and public social media platforms. The objective is to extract relevant citizen opinions and perform sentiment analysis to assess attitudes towards various public services and topics of interest in the pilot cities. The following section describes the planned methods and technologies under consideration, as well as the current status of implementation.

4.2. Content Collection Methods

To gather relevant textual data from diverse sources, the following content collection strategies have been considered or adopted:

4.2.1 Web Crawling

A dedicated web crawler has been implemented to retrieve article text and associated metadata from official municipal websites of the pilot cities at regular intervals. This crawler will:

- Extract main article content (headlines, body text) for later processing.
- Support multilingual extraction (Romanian and Slovak).

This tool is a tailor-made solution, taking into account each pilot city's website structure. The websites analyzed and crawled are the following:

- **Brasov:** https://www.brasovcity.ro/ro/ultimele_noutati/1
- **Martin:** <https://www.martin.sk/dp/vybery=1>

In upcoming phases, the plan is to also extract associated media of the articles (e.g., images).

4.2.2 Social Media Data Acquisition

Publicly available content from official Facebook pages of municipal authorities is considered for inclusion. The planned approach takes into account:

- Scraping or API-based retrieval of public posts and their comments.
- Extraction of user reactions (likes, shares, etc.) as potential additional sentiment signals.

X, formerly known as Twitter, has also been considered for content extraction, but has currently been excluded due to prohibitive API access costs and restrictions.

At this stage, the social media data collection process remains in the design phase, pending final tool/method selection and API access validation, if eventually granted by Meta.

4.3. Multilingual Content Processing and Analysis

So far, there have been two approaches to implementing a multilingual processing pipeline in order to normalize and prepare the collected data for sentiment analysis. Researching which approach (and model) produces the best results has been an ongoing process, with the eventual choice to be decided through preliminary analysis results on citizen opinions or detected trends, provided in the next project phase after data collection and system integration are completed. The expected outcome is that comments on platform topics will be analyzed for positive, negative, or neutral sentiment, and comments on Facebook posts from municipal pages will undergo similar sentiment evaluation. Sentiment trends will be aggregated per topic to detect general public opinion tendencies regarding services, decisions, or events.

4.3.1 Assessing data directly from original language (Romanian, Slovak)

The following models from HuggingFace have been tested with Romanian and Slovak text input passed to a sentiment pipeline, with various outcomes:

- xlm-roberta-large
- cardiffnlp/twitter-roberta-base-sentiment
- nlpstown/bert-base-multilingual-uncased-sentiment
- dumitrescustefan/bert-base-romanian-uncased-v1

Additionally, the following model has been tested as proof of concept for differences between an Romanian text and its English equivalent, which produced inconsistent results for the Romanian text:

distilbert-base-uncased-finetuned-sst-2-english

All of the above models are pre-trained, but have not been further fine-tuned during the implementation phase. This is a task to be considered and potentially undertaken during the next phases of development.

4.3.2 Translating original text to English and subsequently assessing data

Because of the fact that -as of yet- no models fully compatible with Romanian and Slovak have been identified, a different approach has been implemented and tested: Translating the original Romanian/Slovak text into English, and then performing sentiment analysis on the translated text. This technique is further enhanced by processing each sentence separately, so that sentiment analysis produces as accurate results as possible.

The models used to translate the original texts is *Helsinki-NLP/opus-mt-ROMANCE-en* for Romanian and *Helsinki-NLP/opus-mt-sk-en* for Slovak (*Helsinki-NLP/opus-mt-ro-en* has been tested for Romanian too). Translations are then fed to the *paraphrase-multilingual-MiniLM-L12-v2* sentence model in order to encode and extract sentences, and then those sentences are subsequently fed to the *distilbert-base-uncased-finetuned-sst-2-english* sentiment analysis model.

4.4 Integration into Platform

Processed and translated texts will be stored in the platform as articles and/or discussion topics, either manually created or automatically generated from news and social media content. These topics will then serve as a basis for:

- User-driven discussions within the platform, whose comments will also undergo sentiment analysis.
- Automatic sentiment analysis of social media comments related to official municipality posts.

The tools to perform these tasks are made available through an API that uses JWT authentication.

5. Cognitive and Social Agents (T3.4)

This task involves the design of Natural Language Processing (NLP)-based components for clustering and identifying discussion topics, as well as developing automated content moderation mechanisms to assist human-based moderation. These components aim to contribute to a deeper understanding of discussions and improve the safety and relevance of platform interactions. Texts to be processed include topics created on the platform, along with web content collected via the tools described in Chapter 4 (T3.3).

5.1 Categorization and clustering of texts

So far, two methods have been explored:

5.1.1 Categorization through sequence classification

This method first translates the Romanian and Slovak texts using the translation models described in 4.3.2 and subsequently passes the resulting english texts as tokens (with the aid of transformers.AutoTokenizer) to the **cardiffnlp/tweet-topic-large-multilingual** sequence classification model. From there, topic labels are predicted for each text.

5.1.2 Clustering and keywords through a sentence transformer

This method involves using a sentence transformer - so far, the model that has been tested is **paraphrase-multilingual-MiniLM-L12-v2** - and then passing the embeddings created to a clustering algorithm (KMeans module has been selected for that). The final output is a collection of clusters along with their keywords.

The main caveats of this process which need to be researched and addressed are:

- Finding out if it is more reliable to first translate the original content or pass it to the sentence transformer after some Unicode normalization (due to special characters existing in Romanian and Slovak)
- Identifying the appropriate number of topics for clusters
- Using stop words for the vectorizer that identifies keywords
- Assessing the keywords' value (e.g. some keywords may be irrelevant or just common words)

5.2 Automated moderation

The core part of this task is the development of an automated moderator ("bot"), which will ensure that content uploaded to the platform meets community guidelines for safety and appropriateness. All user-submitted posts and comments are submitted to classification in order to detect (among other issues to be considered) hate speech, toxic language and harassment or offensive content. This classification extends to uploaded images as well, which are checked for inappropriate or harmful content using image classifiers.

Moderation workflow is the following:

- Analysis of submitted text and images through an API. At this point, content is uploaded but not yet visible to the platform, as it is pending validation.
- Automatic rejection, flagging, or approval of content based on classifier outputs.

The classifier used for text is **unitary/toxic-bert**, whose behaviour seems the most consistent so far, with other candidates being tested: **Hate-speech-CNERG/dehatebert-mono-english** and **google/shieldgemma-2b**. Text goes through the selected model's `text-classification` pipeline and outputs probabilities of inappropriate content.

For image moderation, the CLIP (dual-encoder model trained on image-text pairs) model `openai/clip-vit-base-patch32` has been selected to evaluate potential inappropriate content. This model produced better results so far than other candidates, the latter being: `google/vit-base-patch16-224`, `google/vit-large-patch16-384` and `openai/clip-vit-large-patch14`. The evaluation process involves resizing the image, setting labels of inappropriate content and setting a probability threshold between 0 and 1. The model then classifies the image, assigning a probability per label and if any label surpasses the threshold, the image is deemed inappropriate.

In order to extract more reliable results from the image classification tool, the solution of fine-tuning the model is being investigated. This will involve using a dataset of images with toxic category labels. The dataset considered for this process is the *LAION NSFW* subset (*laion/laion2B-en-nsfw*).

5.3 Vocabulary Dataset

A dedicated vocabulary dataset is planned to be developed from scratch as part of this task. The purpose of this resource is to identify key terms, phrases, and expressions used by citizens and public authorities in the context of e-government, city services, and local issues. This would help improve the functionalities of clustering, keyword extraction, potential user queries, content tagging and possibly automated moderation (by including known abusive or sensitive terms).

This process would involve gathering a representative collection of texts to mine vocabulary from the platform posts and the content retrieved from the web, as described in Chapter 4. An extra step of human involvement for filtering the resulting vocabulary and fine-tuning it may be necessary as well.

Automated term extraction tools considered for this task are the following:

- TF-IDF (Term Frequency-Inverse Document Frequency)
- RAKE (Rapid Automatic Keyword Extraction)
- YAKE (Yet Another Keyword Extractor)
- KeyBERT (BERT-based keyword extraction),

while vocabulary structuring can be assisted by Protégé, RDFLib and language processing tools.

6. Ontology (T3.5)

As part of the platform's conceptual design phase, a formal ontology-based representation of public sector concepts, actors, services, and procedures is to be defined. This framework will facilitate consistent data modeling, interoperability with external systems, and semantic understanding across components.

The design will prioritize the reuse and extension of well-established, domain-relevant ontologies. Examples to be considered are the following:

- Core Public Service Vocabulary (CPSV), EU ISA² standard for modeling public services and their delivery contexts.
- FOAF (Friend of a Friend), for describing agents such as citizens, civil servants, or organizations.
- Dublin Core & Schema.org, for general-purpose metadata and document description.

- ADMS (Asset Description Metadata Schema), where applicable, for public-sector metadata assets.

The development and evolution of ontologies is expected to rely on a combination of manual modeling and tool-supported management. The use of ontology editors like Protégé, as well as OWL 2 and RDF as the primary knowledge representation formats, in line with W3C recommendations, are under consideration.

Furthermore, the use of semantic annotation to enrich data and content with machine-readable semantics is being researched, in order to associate content (services, processes, user actions) with relevant classes and properties from the ontology and potentially support multilingual annotations.

7. Gamification Incentive Mechanism (T3.6)

ITHACA's task 3.6 aims to develop an incentive mechanism to (i) cluster users based on their participation and to (ii) award points based on the user's contribution to the content (discussions, comments, votings, etc.) on the ITHACA Platform. This kind of incentive mechanism is called gamification which has been defined as '*the use of game design elements in non-game contexts*' (Deterding et al. 2011, p. 9). The underlying principle is to implement specific design features or 'motivational affordances' (Deterding, 2011; Zhang, 2008) as it is usually done in many computer games and in other contexts to make engagement more motivating. Thus, the main objective of gamification is to increase engagement (Kapp, 2012; Villagrasa, Fonseca, Redondo, & Duran, 2014). Even if the term 'gamification' is fairly new (21st century), from a psychological point of view, the underlying principles are basically the same as those of operant conditioning, which was already empirically researched and extensively described in the first half of the 20th century (e.g., Thorndike, 1905, Skinner, 1938). In operant conditioning, the frequency of occurrence of certain behaviours is modified by an association with the addition or removal of reward or aversive stimuli (which leads to four cases: addition of reward, addition of aversive stimuli, removal of reward, removal of aversive stimuli). Gamification usually focuses on the addition of reward (i.e., getting points or badges for showing desirable behaviour) and the removal of reward (i.e., getting minus-points or losing / downgrading badge-ranks for undesirable behavior).

In principle, it seems advisable to avoid reinventing the wheel. Therefore, a state-of-the-art literature analysis was carried out in ITHACA to learn from best-practice examples. This literature review was also guided by the following considerations and research questions:

I. How to transfer a user's extrinsic motivation into intrinsic motivation? In a nutshell: extrinsic motivation is the motivation to do something because of (future) external rewards (or punishments), associated with that activity or behavior. Intrinsic motivation is the motivation to do something because it is the activity itself that is perceived as rewarding (joyful, interesting, important and relevant, etc.).

II. In ITHACA, we are targeting a wide range of users, i.e. also elderly people, IT-inexperienced, etc. Therefore, gamification mechanics that 'work' and are intuitively understandable should be used, even if they might be old-fashioned and not very fancy.

III. The gamification mechanics (e.g., the reward systems and incentive mechanism) should be made transparent and thus, should be explained to the users.

IV. We would like to facilitate 'quality' (e.g., of the comments, discussions, proposals, etc.) rather than 'quantity'. For example, if the game mechanics solely or primarily would take into account the number of comments made by users, this would motivate behaviour that could lead to the platform's content being extensive but also full of spam (and in consequence, it might de-motivate intrinsically motivated users to contribute with content).

ad I.: Self-determination theory as underlying theoretical framework to foster intrinsic motivation

A best-practice theoretical 'framework' is the *Self-determination Theory* (SDT, e.g., Ryan & Deci, 2000; Ryan et al., 2021) that has often been applied as theoretical backbone in gamification research (e.g., Cruz, Hanus, & Fox, 2017; Mekler et al., 2017; Seaborn & Fels, 2015; Sailer et al., 2017; Sailer & Homner, 2020).

Thus, we will give a very brief overview on the SDT:

The SDT consists of six 'mini-theories' (see also <https://selfdeterminationtheory.org/theory/> for an introduction):

1. Cognitive evaluation theory (CET): deals with the relationship between intrinsic motivation and external rewards (as it is the case in gamification). If external rewards have a controlling effect or if they put pressure on the individual to behave in a certain way, they reduce intrinsic motivation according to CET. But if external rewards are informative and provide feedback on certain behaviours, they potentially increase intrinsic motivation.
2. Organismic integration theory (OIT): suggests and describes four different types of extrinsic motivations (external regulation, introjected regulation, identified regulation, and integrated regulation).
3. Causality orientations theory (COT): explores individual differences in the way people motivate themselves in regards to their personality. An example from sports: some people do sports, primarily because they can win a prize (control orientation), others just enjoy it as such (autonomy) and others orient more towards barriers and obstacles of doing sports (impersonal orientation).
4. Basic needs theory (BNT): recognizes three psychological needs related to intrinsic motivation, effective functioning, qualitative engagement and psychological well-being. The first psychological need is autonomy, i.e., the belief that one can determine one's own behaviour and actions. The second psychological need is competence. Competence means that someone is able to work effectively because they have mastered their ability to interact with the environment. The third psychological need proposed in the theory of basic needs is relatedness, which is the need to build strong relationships or bonds with the people around a person.

5. Goal contents theory (GCT): compares the benefits of intrinsic goals with the potential negative outcomes of external goals in terms of psychological well-being.
6. Relationship motivation theory (RMT): analyses the importance of relationships. This theory assumes that high-quality relationships satisfy all three psychological needs described in BNT.

In particular the CET, the OIT and the BNT are considered as highly relevant in the context of gamification.

Conclusions:

In ITHACA, the basic need of autonomy is already fulfilled, as users can decide for themselves whether they want to contribute at all and if so, to what extent. However, in order to facilitate the transition from extrinsic motivation to intrinsic motivation, the gamification mechanics should at least focus on the remaining two basic needs, competence and relatedness. For example, there could be a badge (see also point II below) associated with the concept 'competence'. This badge should be provided, if users apply not only simple actions, such as evaluating contributions / discussion points / comments of others, but actively write a comment / proposal by themselves. Or there could be a badge associated with the concept 'relatedness' for activities that are carried out for or with others (e.g., answering a 'how-to' question from newcomers, working collaboratively and / or cooperatively with others on a proposal, e.g., a suggestion on how a playground could be redesigned). Both of these associated badges require skills in a broader sense (writing skills, argumentation, information search, creativity, etc.) - but since some users may not have these skills or may not be confident in them, there should also be a badge that is fairly independent of the badges and associated skills mentioned above. One possibility would be to basically recognize activity as such, also including more 'passive' activities (e.g., how often someone accesses the platform to keep up to date etc.), as well as the rating of posts with thumbs up or thumbs down or participation in votes. To sum up, based on the SDT we suggest having three badges, one that aims to foster competence, one that aims to foster relatedness, and one that aims to foster activity. In line with the CET, we would like to focus on 'rewards' only (i.e., the user gets points or badges in case of desirable activities and behaviors) rather than including also 'punishments' (such as that a user loses points or badges in case of undesirable activities and behaviors).

ad II.: Best-practice examples of gamification mechanics that work and are intuitively understandable

Based on several meta-analyses on gamification (e.g., Subhash & Cudney, 2018; Sailer & Homner, 2020; Oprescu, Jones, & Katsikitis, 2014; Koivisto & Hamari, 2019; Johnson et al., 2016) that usually focus on other domains, such the educational or work context (which is however, not that relevant because the underlying principles and gamification mechanics should be transferable to the context 'participation in an online platform on participatory democracy'), the following best-practice gamification mechanics / elements and 'types of rewards' have been identified:

- **Trophies** (e.g., De-Marcos et al., 2014),
- **Badges** (e.g., De-Marcos et al., 2014; Villagrasa et al, 2014; Yildirim, 2017; Dias, 2017 ; Sailer et al., 2017),

- **Points** (e.g., Villagrasa et al., 2014; Markopoulos et al., 2015; Yildirim, 2017; Dias, 2017),
- **Levels** (e.g., De-Marcos et al., 2014; Markopoulos et al., 2015),
- **Leaderboard** (e.g., De-Marcos et al., 2014, Leaning, 2015; Yildirim, 2017; Dias, 2017; Sailer et al., 2017),
- **Avatars and customization** (e.g., Villagrasa et al, 2014; Sailer et al., 2017),
- **Achievements and quests** (e.g., Villagrasa et al., 2014; Markopoulos et al., 2015).

A holistic gamification framework called Octalysis [1] (see also Weber, Grönwald & Ludwig, 2022) should also be mentioned since it combines a wide range of gamification mechanics and underlying principles from behavioral science.

Conclusions:

As can be seen from this far from complete list, the possibilities for incorporating different gamification elements are manifold. However, as mentioned above, even if there would be quite innovative and novel possibilities, the gamification principles should work and should still be as intuitive as possible. The Octalysis gamification framework seems either too complex or – if we would reduce its complexity and holistic approach too much – we would come up with a similar solution as described here: We would suggest awarding 3 badges (i.e., ‘medals’) for the associated concepts mentioned under point I, simply in bronze, silver and gold; whereas the achievement of these badges should become increasingly difficult (bronze should be very easy and quick to achieve, gold very difficult and after some effort). Thus, the ‘badge-colors’ also reflect different levels, whereas these levels should be increasingly difficult to achieve. For example, for a bronze badge, a user may be required to collect 10 points, for a silver badge 100 points and for a golden badge 1000 points. Thus, we apply in a rather simple way at least ‘the classics’: badges, points and levels. To also motivate younger users and/or advanced users that are more familiar with social media, social platform and gaming, etc., could be offered. A leaderboard (e.g., a list of users who have already achieved three gold badges) could also be offered, but it should be up to the user whether he or she wants to be listed there and whether or not to show his or her badges and points to other users.

ad III.: Making the ‘rules’ on how to get what kind of badge explicit

Rather self-explanatory, but users should (if they wish) be informed about the ‘rules’ by means of ‘how-to’ information, i.e., how to achieve which badges and points and which types of activities are rewarded and to what extent.

ad IV.: Avoidance of undesirable side effects of the gamification approach

In principle, a user could achieve all three gold badges if he or she performs the same easy (and low-scoring) activities often enough (e.g., writing one-word comments, or just logging in and out several times a day, or similar), unless further mechanics are added that do not over-reward these types of behaviors. Thus, in order to achieve a silver or golden badge, also more high-scoring or ‘harder’ activities should be required.

The following two tables list activities that should be rewarded with points and corresponding badges. We differentiate between activities (or ‘missions’) that can be completed or rewarded only once (i.e.

single acquisition badges; see Table 2) and activities that can be rewarded repeatedly (i.e. repetitive badges; see Table 3).

Table 2 List of activities (clustered into easy, medium, and difficult) that can be carried out (and thus, rewarded by badges) only a single time

Badges for the acquisition of single activities	
Bronze	<ul style="list-style-type: none"> *Create your 1st post (Competence) *Create your 1st comment (Competence) *Navigate/Engage through X component (X being chatbot, argument visualization, simplification tool, TSD, evaluation tools, etc. (Activity)
Silver	<ul style="list-style-type: none"> *Update “About Me” question. (Relatedness) *Complete all Bronze level missions (Competence)
Gold	<ul style="list-style-type: none"> *Have your proposal passed on a civic topic (Relatedness) *Have your argument be the most voted one in a civic topic (Relatedness)

Table 3 List of activities worthy of reward, their ‘difficulty level’, and the badges (or types of basic needs they should facilitate

Badges for the acquisition of repeated activities			
	Badge on Competence	Badge on Relatedness	Badge on Activity
Easy (1-10 Points); required to reach bronze status	<ul style="list-style-type: none"> *The user evaluates a comment of another user (e.g., by thumbs-up). *The user participates in a voting. 	<ul style="list-style-type: none"> *The user associates him-or herself with other users (i.e. getting ‘friends’) *The user forwards a comment or proposal to others who might be interested in it. 	<ul style="list-style-type: none"> *Number of logins *Amount of time spent on the platform. *Browse through the 5 most popular X (arguments, posts, etc.)

			*Browse through the 5 most recent X (arguments, posts, etc.)
Medium (11-25 Points), required to reach silver status	<p>*The user writes a reply to a comment of another user.</p> <p>*The user writes a comment.</p> <p>*The user gets positive evaluations from other users for his/her comment.</p> <p>*The user who replies with a constructive comment to another users proposal receives an approval-rating (thumbs-up) by the the user who wrote the proposal (by this, the proposal-writer could be rewarded because he or she is 'allowed' to reward associated contributions / comments of other users that might even improve the initial proposal, and this would reward 'constructive' comments/ replies)</p>	*The user answers a 'how-to' question of another (new) user.	<p>*Make X posts with a positive feedback ratio (week duration)</p> <p>*Engage with X component (e.g. start a discussion/voting in argument visualization with positive feedback)</p>
Difficult (26-50 Points), required to reach gold status	*The user writes a proposal (e.g. ideas for the city, such as how public transport could be improved).	*The user initiates a (successful) voting; whereas successful means that a lot of other users participate in the voting.	

	<p>*The user does some research on city-related solutions and approaches from another (foreign) European city. For example: A user visits another European country / city. He or she sees a nice playground. He or she makes a photo and posts it, together with a description on why he or she thinks that this could be adapted to his/her hometown as well.</p>	<p>*The user writes a proposal (see also cell at the left) together with others - i.e. collaboratively and cooperatively. Up to a certain group size, e.g. 5 users, the points should be equally distributed and as high as if the user would do it alone. Above that, the 'individual reward' (i.e. the number of points a single user of the group gets) should be reduced accordingly.</p>	
--	--	---	--

8. Public AI Register (T3.7)

At the present stage, the development of the Public AI Register pages has not commenced. The following process outlines the intended steps for designing and implementing these pages, ensuring alignment with transparency, accountability, and public accessibility requirements:

8.1. Requirements Gathering

- Identification of legal, ethical, and technical requirements, including compliance with the EU AI Act, GDPR, and relevant local regulations.
- Consultation with pilot cities regarding the translation of documentation and any region-specific content needs.

8.2. Information Architecture Design

Considerations regarding the main content sections of the Public AI Register include:

- An overview of the AI tools
- Details on decisions and assumptions in the development and deployment of AI tools
- Risk assessments and mitigation measures, along with caveats and issues to consider
- Decommissioning strategies.

Apart from the above, a key issue to consider is the development of a navigation structure that ensures ease of access for diverse user groups.

8.3. Wireframing and Prototyping

A proposed approach to be discussed with experts on webpage design on how to prototype the register is to create wireframes demonstrating the page layout and content hierarchy, multilingual content support and sections for user-friendly descriptions and downloadable documents.

8.4. Implementation Planning

During the implementation planning, the technology stack and tools will be determined. Planning for frontend (and backend if needed) integration with the main platform will also take place.

8.5. Accessibility and Usability Testing

One thing to consider is the preparation for compliance with Web Content Accessibility Guidelines (WCAG) 2.1. Furthermore, potential scheduling of user testing sessions in the pilot cities to validate clarity and accessibility should be considered as well (to be discussed with partners and pilot cities).

9. Argument Visualization (T3.8)

Argument Visualization (AV) involves the graphical representation of arguments and reasoning to enhance understanding and communication. The AV component in ITHACA helps users organize information and reason systematically, promoting clearer thinking and more effective decision-making. In ITHACA, the AV component will display a visual representation of citizens' opinions on various ideas suggested by others, ranging from agreement to disagreement. It will also include a ranked list of advantages and disadvantages. This component will use a pie chart to illustrate the various ideas put forth by citizens and the extent to which these ideas are embraced by others. A pie chart is chosen because the number of proposed solutions for a specific issue might be too low to effectively use a heat map. Consequently, the most widely accepted ideas for each topic will be clearly visible to all.

In Figures 3 and 4, a draft design for the initial release of the argument visualization component is presented, based on the currently available version of the ITHACA platform's user interface. By selecting the "Arguments" tab, users can access a list of public issues, posted by users with elevated privileges (such as municipal users from cities testing the platform). In this list, a short description of each issue will appear, as depicted in Figure 3.

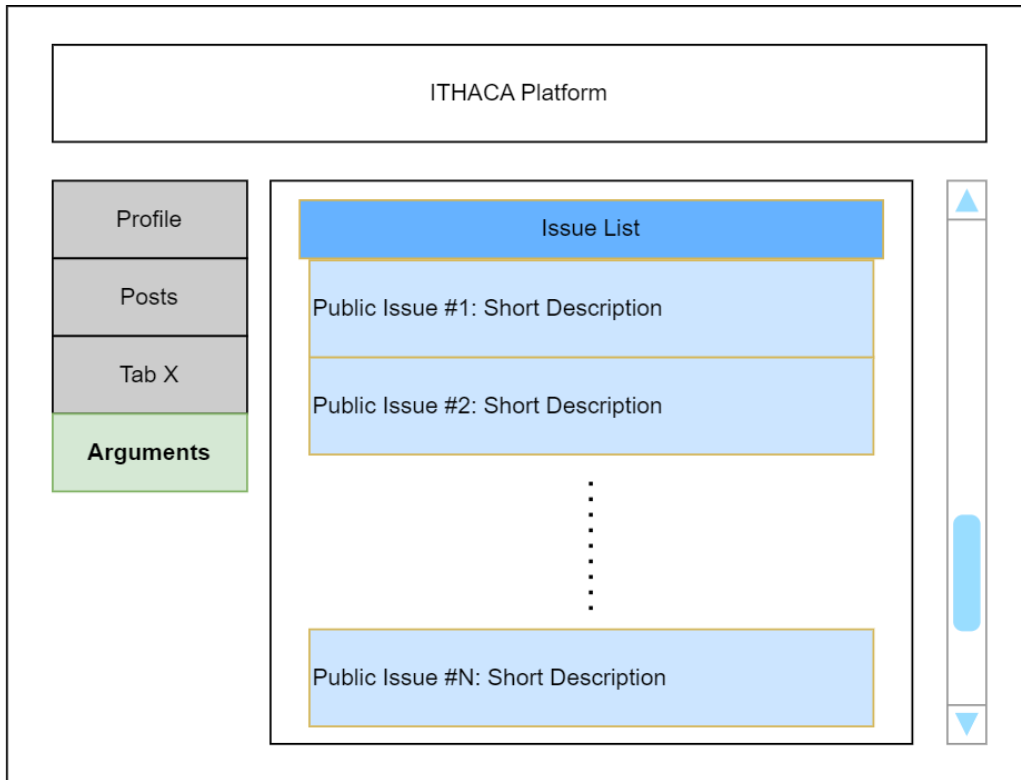


Figure 3 The screen appears in the ITHACA user interface, upon pressing the “Arguments” tab.

The user can choose any issue from the list by clicking on it, which will bring up a new screen (see Figure 4). This screen will display a detailed description of the issue along with a list of proposals submitted by other citizens or users on how to solve the issue. Every citizen also has the right to contribute a new proposal or to add a new argument in support of or against an existing proposal for a particular issue, as can be done through the "Add Argument" button shown in Figure 4.

The ITHACA approach of the AV component (Figure 4) overcomes the disadvantages of a ranked pros-cons list. The latter can lead to biased decisions under certain conditions, for example when users omit critical disadvantages or overemphasize certain benefits, or when citizens use emotionally charged language in describing the pros and cons. This approach will help mitigate potentially toxic comments arising from strong disagreements with particular opinions and prevent any user bias in voting for ideas. It ensures that all citizens' proposals regarding a public issue are treated equally and objectively, making sure that every voice is heard.

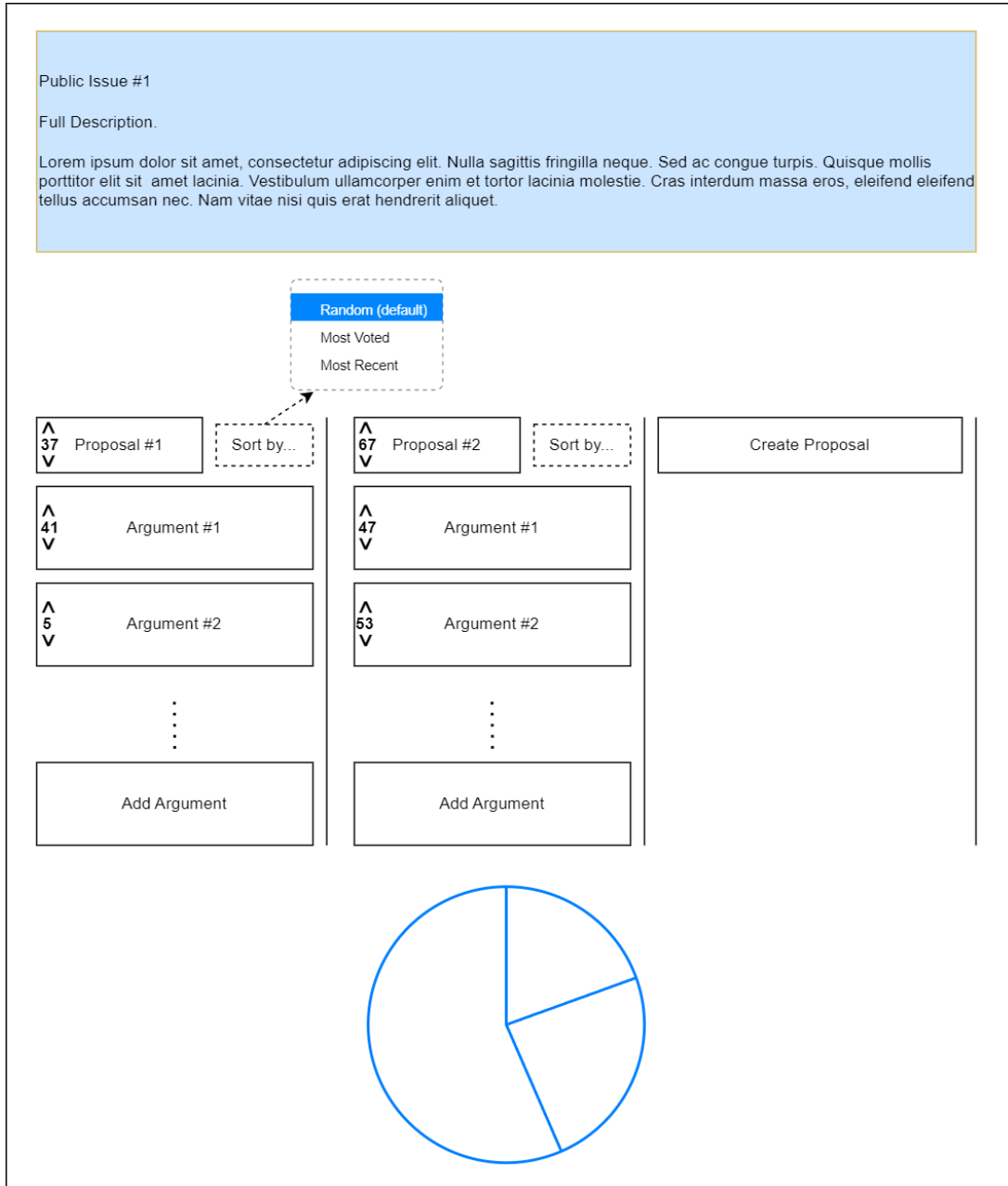


Figure 4 A first design of the window appearing in the ITHACA user interface, upon selecting an issue from the “Argument” tab (Figure 3)

Proposals are ideas or solutions addressing public issues added by citizens in the AV component. For instance, a public issue may be how to utilize an open-space urban area, and the proposed idea may be to build a parking lot or a playground. Other citizens will be able to post their arguments under each proposal to state their agreement or disagreement. A new proposal/solution to an issue can be created by pressing the button “Create Proposal” (see Figure 4).

Citizens can vote for or against proposals or arguments depending on their agreement or disagreement. The tally of votes for both the proposals and arguments will be displayed alongside them, as illustrated in Figure 4. Thus, each proposal and argument will have an accumulated number of votes that are calculated as a function of #upvotes - #downvotes. Before the user casts their vote, the proposals and arguments will be displayed in a random order to not prejudice users over which proposal to vote for and avoid bias towards already popular solutions/arguments. Moreover, the user will have the option to sort the arguments of a proposal by the most voted or most recent ones only after they have casted their vote (“Sort by...” button as illustrated in Figure 4).

The pie chart located at the bottom of the tool visually represents the proposals. The size of the wedges depends on the sum of the votes across proposals. Therefore, highly rated proposals appear as larger segments, while proposals with fewer votes are shown as smaller segments. This chart enables users to quickly understand which proposals are well accepted and which are not favored by the community.

Overall, the argument visualisation serves as a community tool designed to engage citizens in discussions about public issues. It gathers diverse opinions and encourages citizen participation by allowing them to contribute their own proposals or arguments related to existing ones. The graphical representation of proposals, arguments, and votes should reinforce the motivation to participate. In contrast to existing research of argument mining and visualization, this approach does not analyse the semantic structure of arguments, but emphasizes community involvement and participatory aspects.

10. Trust and Security Infrastructure (T3.9)

10.1. Security Architecture and Trust Mechanisms

The platform’s security architecture is designed to ensure the confidentiality, integrity, and availability of user data and platform services. The approach is multi-layered and leverages both open-source and enterprise-grade security technologies.

10.1.1 Authentication and Authorization

The platform employs Keycloak, an open-source identity and access management solution, as the central mechanism for handling user authentication, and Single Sign-On (SSO). This ensures that users, services, and APIs are properly authenticated and that access rights are enforced according to organizational policies.

10.1.2 Endpoint Protection

FortiClient, an enterprise security suite, is used to secure endpoints connecting to the platform infrastructure, providing malware protection, VPN capabilities, and compliance enforcement at the client level.

10.1.3 API Security:

All communication between platform components, as well as with external services, is secured via HTTPS (TLS encryption), ensuring data confidentiality and integrity in transit. API-level security includes verification of JWT tokens for every API call, and role permission enforcement based on claims embedded in the JWT.

10.1.4 Trust Mechanisms

Trust within the platform is established by:

- Centralized identity and access control via Keycloak,
- Validated cryptographic tokens for secure API communication,
- Encrypted data exchanges between microservices,
- Secure onboarding of third-party services or data providers subject to identity verification and security assessments.

Further extensions of the trust framework are under consideration for future development phases.

10.2 Adherence to WS-Security and Interoperability Standards

While the platform is primarily based on RESTful APIs secured via HTTPS and JWT tokens, future integration with services or external systems may require SOAP-based web services. To ensure such interoperability:

- **WS-Security Compliance:** If SOAP-based services are adopted, adherence to the WS-Security standard is under research to be supported by:
 - XML Signature and XML Encryption,
 - Security token handling (e.g., SAML 2.0),
 - Timestamp and message integrity enforcement.
- **REST API Standards:** current REST APIs are designed with a commitment to:
 - Ensuring consistency, scalability, and ease of integration with third-party services (such as public registries or external semantic services),
 - Adoption of common data formats (e.g., JSON) to enable semantic interoperability. Interoperability remains a guiding principle for platform development, particularly as integration with e-government systems or AI transparency modules progresses.

10.3 Glossary Development and Encryption Methods

A dedicated security glossary is under consideration to be developed in line with overall platform ontology and terminology work, ensuring clear definitions of security and trust terms, consistency across all documentation, API specifications and end-user guidance, and alignment with respective standards. This glossary would help support both internal developer communication and external reporting (e.g., to regulatory bodies or public AI registers).

The platform applies TLS (HTTPS) to secure all data in transit between users, services, and external APIs. JWT tokens are cryptographically signed to prevent tampering and to ensure authenticity of requests, taking into account the sensitivity of stored information and applicable regulatory requirements.

11. Personal Information Management System (T3.10)

As part of Task 3.10, the development of the Personal Information Management System (PIMS) is intended to address the principles of personal data protection, user autonomy, and transparency in data processing. The design and implementation of this system will rely on the best practices identified in Task 1.6 and the specific user requirements gathered in Work Package 2 (WP2).

At the time of this deliverable, PIMS is at the stage of deliberation and requirement collection, as the final specifications depend on the coordinated integration with the platform's AI components and the central user management system. Additionally, conclusions and guidelines from Task 1.6 are expected to influence the feature set and implementation priorities.

PIMS is expected to provide users with control over their personal data, enabling them to modify or revoke permissions of their data at any time through a dedicated user interface. The types of personal data to be collected, purposes of storage/processing and the duration of storage/processing will be made known to users through this system, in accordance with applicable data protection regulations. The user interface will be made available in all platform languages.

12. System Integration, Testing, and Deployment (T3.11)

The platform development adopts a modular architecture with considerations for service independence, scalability and maintainability (as defined in section 3.3) in order to facilitate:

- Independent development and deployment of AI tools, data management services, user interface components, and backend APIs.
- Easier updates and maintenance, without requiring full system redeployment.

Containerization benefits:

- All major platform components (e.g., AI services, PIMS, semantic indexing services, web interfaces) are expected to be containerized using Docker or similar technologies to enable

portable, reproducible deployment across different environments (development, testing, production).

Additionally, orchestration and service management are achieved through utilizing Kubernetes. This leads to easier Management of container lifecycles, auto-scaling, load balancing, monitoring and fault tolerance.

Sections 3.3.7 and 3.3.8 provide more insight to aspects of containerization and orchestration.

Platform integration includes:

- APIs to connect user management and PIMS with AI processing modules.
- Data exchange protocols to support semantic indexing, annotation, and search functionalities.
- Secure inter-service communication and user context sharing mechanisms.

Role of CI/CD Practices in Development

To ensure consistent, secure, and rapid development, Continuous Integration and Continuous Deployment (CI/CD) practices will be implemented as follows:

- Version Control Integration: All codebases (backend, frontend, AI models) are version-controlled using Git repositories
- Automated Testing Pipelines: CI pipelines will automatically execute any relevant unit tests and integration tests.
- Automated Build and Deployment Pipelines: CD pipelines will build application containers automatically, deploy to staging and production environments after successful testing and support rollback mechanisms in case of failed deployments.
- Dependency Management: Model files (for AI tasks), ontology datasets, and semantic resources will also be integrated into the CI/CD process to ensure they are versioned, reproducible, and tested as part of the deployment. Additionally, persistence across deployments for any training datasets and subsequent trained models will be managed in an effective way.

Scalability Features

The deployment infrastructure will enable horizontal scaling (adding more instances) of web services, APIs, and AI tools. Additionally, supporting elastic resource allocation for computationally intensive components such as sentiment analysis and semantic reasoning modules is to be further researched.

Security Considerations

Secure communication between all platform components is achieved by utilizing encrypted channels (TLS/SSL) to prevent data interception. Furthermore, a role-based access control (RBAC) system will be implemented to ensure that only authorized components and users can access sensitive services

and data. Any AI services processing personal or sensitive content will enforce strict authentication and authorization measures. The deployment will be aligned with GDPR and other relevant data protection regulations. Also, the PIMS module will play a central role in ensuring that data processing aligns with user consents and legal requirements. CI/CD pipelines are intended to incorporate automated security scanning of dependencies and container images to detect and mitigate potential vulnerabilities before deployment.

Finally, one key aspect of security and privacy that has been carefully considered throughout the development process is the ability of the platform to self-host all AI models and tools. This ensures that data exchange remains inside the platform's ecosystem and that no communication with 3rd party providers is being made during the use of said tools.

13. Conclusions

Deliverable D3.1 marks a pivotal step in the evolution of the ITHACA platform, setting the groundwork for a secure, inclusive, and ethically-aligned civic participation ecosystem driven by artificial intelligence. It consolidates the technical, architectural, and conceptual foundations necessary for building a user-centric digital space where citizens can engage meaningfully with democratic processes.

Through a modular and hybrid system architecture—integrating microservices and event-driven messaging—the ITHACA platform ensures flexibility, resilience, and scalability across diverse use cases and city contexts. The comprehensive definition of technical requirements, drawn from real-world pilot scenarios and legal-ethical assessments, establishes a robust development blueprint. It balances cutting-edge AI capabilities—such as automated moderation, multilingual sentiment analysis, and argument visualization—with safeguards for user privacy, transparency, and accountability.

The deliverable also reflects ITHACA's commitment to human-centric design. The incorporation of gamification principles grounded in psychological theory, the creation of a transparent Public AI Register, and the development of tools for data simplification and personalization all aim to enhance user experience and foster democratic engagement across different population segments.

Key technological components—including sentiment extraction, NLP-powered agents, ontology-based semantic reasoning, and a trust-enabling security infrastructure—have been specified, prototyped, or partially implemented, providing the basis for iterative refinement. Integration strategies and continuous deployment workflows ensure that upcoming versions of the platform can evolve seamlessly, informed by ongoing testing, user feedback, and co-creation activities in the pilot cities of Braşov and Martin.

Looking ahead, the outcomes of D3.1 will feed directly into Deliverable D3.2, which will focus on advanced integration, testing, and iterative improvement of platform functionalities. D3.3 will consolidate a refined release aligned with long-term impact goals. The current deliverable thus serves as both a reference point and a springboard, ensuring that the ITHACA platform continues to develop in a way that reinforces democratic values, supports informed public discourse, and protects fundamental rights in an increasingly AI-mediated civic space.

14. References

- Chen, L. (2010). Event-driven architectures in distributed systems.
- Cruz, C., Hanus, M. D., & Fox, J. (2017). The need to achieve: Players' perceptions and uses of extrinsic meta-game reward systems for video game consoles. *Computers in Human Behavior*, *71*, 516-524.
- Deci, E. L., & Ryan, R. M. (Eds.). (2004). Handbook of self-determination research. University Rochester Press.
- De-Marcos, L., Domínguez, A., Saenz-de-Navarrete, J., & Pagés, C. (2014). An empirical study comparing gamification and social networking on e-learning. *Computers & education*, *75*, 82-91.
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In A. Lugmayr (Ed.), Proceedings of the 15th International Academic Mindtrek Conference:Envisioning Future Media Environments (pp. 9–15). New York: ACM.
- Dias, J. (2017). Teaching operations research to undergraduate management students: The role of gamification. *The International Journal of Management Education*, *15*(1), 98-111.
- Fowler, M. (2015). "Microservices." Available at: <https://martinfowler.com/microservices/>
- Johnson, D., Deterding, S., Kuhn, K. A., Staneva, A., Stoyanov, S., & Hides, L. (2016). Gamification for health and wellbeing: A systematic review of the literature. *Internet interventions*, *6*, 89-106.
- Kapp, K. M. (2012). The gamification of learning and instruction: game-based methods and strategies for training and education. John Wiley & Sons.
- Koivisto, J., & Hamari, J. (2019). The rise of motivational information systems: A review of gamification research. *International journal of information management*, *45*, 191-210.
- Markopoulos, A. P., Fragkou, A., Kasidiaris, P. D., & Davim, J. P. (2015). Gamification in engineering education and professional training. *International Journal of Mechanical Engineering Education*, *43*(2), 118-131.
- Mekler, E. D., Brühlmann, F., Tuch, A. N., & Opwis, K. (2017). Towards understanding the effects of individual gamification elements on intrinsic motivation and performance. *Computers in human behavior*, *71*, 525-534.
- Oprescu, F., Jones, C., & Katsikitis, M. (2014). I PLAY AT WORK—ten principles for transforming work processes through gamification. *Frontiers in psychology*, *5*, 14.
- Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, *55*(1), 68-78.

- Ryan, R. M., Deci, E. L., Vansteenkiste, M., & Soenens, B. (2021). Building a science of motivated persons: Self-determination theory's empirical approach to human experience and the regulation of behavior. *Motivation Science*, 7(2), 97-110.
- Sailer, M., & Homner, L. (2020). The gamification of learning: A meta-analysis. *Educational psychology review*, 32(1), 77-112.
- Sailer, M., Hense, J. U., Mayr, S. K., & Mandl, H. (2017). How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in human behavior*, 69, 371-380.
- Seaborn, K., & Fels, D. I. (2015). Gamification in theory and action: A survey. *International Journal of human-computer studies*, 74, 14-31.
- Skinner, B. F. (1938). *The behavior of organisms: An experimental analysis*. New York: Appleton-Century.
- Subhash, S., & Cudney, E. A. (2018). Gamified learning in higher education: A systematic review of the literature. *Computers in human behavior*, 87, 192-206.
- Thorndike, E. L. (1905). *The elements of psychology*. New York: A. G. Seiler.
- Villagrasa, S., Fonseca, D., Redondo, E., & Duran, J. (2014). Teaching case of gamification and visual technologies for education. *Journal of Cases on Information Technology (JCIT)*, 16(4), 38-57.
- Weber, P., Grönewald, L., & Ludwig, T. (2022). Reflection on the Octalysis framework as a design and evaluation tool. In *GamiFIN* (pp. 75-84).
- Yildirim, I. (2017). The effects of gamification-based teaching practices on student achievement and students' attitudes toward lessons. *The Internet and Higher Education*, 33, 86-92.
- Zhang, P. (2008). Technical opinion Motivational affordances: reasons for ICT design and use. *Communications of the ACM*, 51(11), 145-147.

Additional references

Apache Kafka Documentation, <https://kafka.apache.org/documentation/>

Docker Engine Documentation, <https://docs.docker.com/engine/>

Hugging Face Documentation, <https://huggingface.co/docs>

Keycloak Documentation, <https://www.keycloak.org/documentation>

Ubuntu Documentation, <https://ubuntu.com/documentation>