



# TITAN

## DELIVERABLE REPORT D4.1

### DELIVERABLE

SUBMISSION DATE	NAME OF THE DELIVERABLE	WORK PACKAGE
27.9.2024	D4.1: TITAN Citizen Intelligent Coaching Platform Architecture and 1st release incl. Training Datasets	WP4
NATURE	AUTHOR(S)	LEAD BENEFICIARY
Public	Marco Cipolla (ENG), Massimo Magaldi (ENG), Spyros Papafragkos (ATC), Georgios Petasis (NCSR-D), Ville Ollikainen (VTT), Arttu Lämsä (VTT), Sara Nikula (VTT), Outi-Marja Latvala (VTT), Marcin Woźniak (SWC)	ENG

### PROJECT DETAILS

PROJECT ACRONYM	TITAN	GRANT AGREEMENT	101070658
CALL IDENTIFIER	Human-01-27	PROJECT DURATION	01.09.2022 – 30.09.2025
PROJECT OFFICER	Peter Friess	PROJECT COORDINATOR	Engineering S.p.A.

## QUALITY CONTROL ASSESSMENT

VERSION	DATE	DESCRIPTION	NAME	ORG
V0.1	12/01/2024	Table of Content	Marco Cipolla	ENG
V0.2	02/02/2024	First draft of all sections	All authors	ENG, ATC, NCSR-D, VTT, SWC
V0.3	26/02/2024	Second draft of all sections	All authors	ENG, ATC, NCSR-D, VTT, SWC
V1.0	15/3/2024	Internal reviewed version	Massimo Magaldi	ENG
V1.1	19/04/2024	Revised version based on Living Labs feedback and insights	Marco Cipolla, Georgios Petasis, Arttu Lämsä	ENG, NCSR-D, VTT
V1.2.0	13/09/2024	Revised version based on pilot feedback and insights	Marco Cipolla, Georgios Petasis, Arttu Lämsä, Marcin Woźniak	ENG, NCSR-D, VTT, SWC
V1.2.1	27/9/2024	Official review & feedback	Marcin Woźniak	SWC
Final	30/9/2024	Final version after revision and quality check	Marco Cipolla Massimo Magaldi	ENG

## DISCLAIMER

The opinions stated in this report reflect the opinions of the authors and do not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein. All intellectual property rights are owned by the TITAN consortium members and are protected by the applicable laws. Except where otherwise specified, all document contents are: “©TITAN Project - All rights reserved”. Reproduction is not authorised without prior written agreement. The commercial use of any information contained in this document may require a license from the owner of that information.

## STATEMENT OF ORIGINALITY

This Deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

## ACKNOWLEDGEMENT

This document is a deliverable of the TITAN project, which has received funding from the European Union’s Horizon 2020 Programme under Grant Agreement (GA) #101070658 and by UK Research and innovation under the UK governments Horizon funding guarantee grant numbers 10040483 and 10055990.

**TABLE OF CONTENTS**

<b>EXECUTIVE SUMMARY .....</b>	<b>8</b>
<b>1 INTRODUCTION.....</b>	<b>10</b>
1.1 GENERAL DESCRIPTION OF THE FRAMEWORK COMPONENTS AND THEIR RELATIONS .....	10
1.2 RELATED WORK PACKAGES AND TASKS.....	12
<b>2 MAPPING OF USER REQUIREMENTS INTO THE TITAN PLATFORM COMPONENTS.....</b>	<b>13</b>
2.1 ITERATIVE REQUIREMENT MANAGEMENT .....	13
2.2 CRITICAL THINKING ASSESSMENT .....	15
2.3 SOCRATIC CHATBOT, DISINFORMATION SIGNALS, MICROLESSONS & INCENTIVES.....	15
2.4 PROPAGATION IMPACT ASSESSMENT .....	15
2.5 COLLABORATION ENVIRONMENT.....	16
<b>3 TECHNICAL REQUIREMENTS.....</b>	<b>17</b>
3.1 COMMON DESIGN CONCEPTS.....	17
3.1.1 <i>Privacy by Design</i> .....	17
3.1.2 <i>Microservice Architecture</i> .....	17
3.2 GENERAL REQUIREMENTS.....	18
3.2.1 <i>Reliability</i> .....	18
3.2.2 <i>Availability</i> .....	18
3.2.3 <i>Monitoring</i> .....	19
3.2.4 <i>Data and System</i> .....	19
3.2.5 <i>Security</i> .....	19
3.2.6 <i>Auditability</i> .....	20
3.2.7 <i>Deployability</i> .....	21
<b>4 TITAN PLATFORM COMPONENTS.....</b>	<b>22</b>
4.1 USER CONTENT .....	22
4.2 CRITICAL THINKING ASSESSMENT .....	22
4.3 DISINFORMATION SIGNAL DETECTION .....	22
4.3.1 <i>Hate Speech, Offensive Language &amp; Clickbait Binary Classification Models</i> .....	22
4.3.2 <i>Multi-Label Hate Speech Detection model</i> .....	23
4.3.3 <i>Logical Fallacy Detection with Large Language Models</i> .....	24
4.4 SOCRATIC CHATBOT .....	26
4.5 PARAPHRASER.....	29
4.6 MICROLESSONS.....	29
4.7 PROPAGATION IMPACT ASSESSMENT .....	29
4.7.1 <i>Propagation impact assessment models</i> .....	30
4.8 COLLABORATION ENVIRONMENT.....	32
<b>5 NEW DATASETS CREATION AND CURATION .....</b>	<b>35</b>
5.1 DISINFORMATION SIGNAL DATASETS.....	35
5.2 DISINFORMATION SIGNAL: HATE SPEECH & OFFENSIVE LANGUAGE BINARY CLASS DATASETS .....	36
5.3 DISINFORMATION SIGNAL: MULTI-LABEL HATE SPEECH DETECTION DATASET .....	37
5.4 DISINFORMATION SIGNAL: CLICKBAIT BINARY CLASS DATASET .....	38
5.5 AUGMENTED LOGICAL FALLACY DATASET FOR LARGE LANGUAGE MODELS.....	39
5.6 SOCRATIC DIALOGUES DATASET .....	40
5.7 ARGUMENTATION MINING DATASET .....	41
5.8 PROPAGATION IMPACT ASSESSMENT DATASET .....	43
<b>6 DOMAIN MODELLING .....</b>	<b>44</b>
6.1 UBIQUITOUS LANGUAGE .....	44
6.2 USE CASES AND USER ROLES.....	45
6.2.1 <i>User Management</i> .....	45
6.2.2 <i>Critical Thinking Assessment Management</i> .....	46
6.2.3 <i>Socratic Conversation Management</i> .....	46

6.2.4	<i>Microlesson Management</i> .....	47
6.2.5	<i>Propagation Impact Assessment Management</i> .....	47
6.2.6	<i>Collaboration Environment Management</i> .....	47
6.3	DATA MODEL.....	48
6.4	BOUNDED CONTEXTS .....	51
<b>7</b>	<b>FRAMEWORK ARCHITECTURE .....</b>	<b>51</b>
7.1	OVERVIEW.....	52
7.2	API GATEWAY.....	52
7.3	MICROSERVICES .....	53
7.3.1	<i>User Management Microservice</i> .....	53
7.3.2	<i>User Content Microservice</i> .....	53
7.3.3	<i>Critical Thinking Assessment</i> .....	53
7.3.4	<i>Socratic Chatbot Microservice</i> .....	53
7.3.5	<i>Microlessons Microservice</i> .....	55
7.3.6	<i>Propagation Impact Assessment Microservice</i> .....	55
7.3.7	<i>Collaboration Environment Microservice</i> .....	55
7.4	MESSAGING LAYER .....	55
7.5	AUDITING AND LOGGING.....	56
7.6	SYSTEM MONITORING .....	56
7.7	ADHERENCE TO PRIVACY-BY-DESIGN PRINCIPLES .....	56
<b>8</b>	<b>TITAN PLATFORM GRAPHICAL USER INTERFACES.....</b>	<b>57</b>
8.1	INITIAL MOCK-UPS .....	57
8.2	TITAN PLATFORM FIRST RELEASE UI .....	64
8.2.1	<i>CT Assessment</i> .....	65
8.2.2	<i>Microlessons</i> .....	66
8.2.3	<i>Socratic Chatbot</i> .....	68
8.2.4	<i>Propagation Impact Assessment (PIA)</i> .....	72
8.2.5	<i>Collaboration</i> .....	74
<b>9</b>	<b>TITAN FRAMEWORK INFRASTRUCTURE .....</b>	<b>76</b>
9.1	SOFTWARE INFRASTRUCTURE .....	77
9.2	NETWORKING .....	78
<b>10</b>	<b>CONTINUOUS INTEGRATION AND DELIVERY .....</b>	<b>79</b>
<b>11</b>	<b>CONCLUSIONS .....</b>	<b>80</b>
<b>12</b>	<b>REFERENCES .....</b>	<b>81</b>
<b>13</b>	<b>ANNEX 1: SOCRATIC DIALOGUE FLOWCHARTS.....</b>	<b>82</b>

## LIST OF FIGURES

Figure 1. TITAN Platform components.....	11
Figure 2. The Gradio Demo Platform.....	23
Figure 3. Rule-based architecture of Rasa Platform taken from RASA documentation.....	27
Figure 4. User input example in Rasa. Example taken from RASA documentation.....	27
Figure 5. Outline of the system communication between the API backend and the rule-based part.....	28
Figure 6. First version of the TITAN Socratic Chatbot UI.....	28
Figure 7. Latest version of the Socratic Chatbot UI.....	29
Figure 8. PIA component and its connections to TITAN platform.....	30
Figure 9. - Interactive argument map with claims, sources and relationships.....	33
Figure 10. - Input box for constructing a supporting argument.....	33
Figure 11. Reusage of previous claims (theses) in input box.....	34
Figure 12. The annotation process in the Ellogon annotation platform.....	42
Figure 13. User Roles.....	45
Figure 14. TITAN Data Model version 1.0.....	48
Figure 15. Entities for Critical Thinking.....	49
Figure 16. Entities for Socratic Conversation.....	50
Figure 17. Entities for Propagation Impact Assessment and Collaboration.....	50
Figure 18. Entities for Microlesson.....	51
Figure 19. TITAN Platform Architecture.....	52
Figure 20. General scheme of Socratic messages management.....	54
Figure 21. TITAN landing page.....	58
Figure 22. Critical Thinking Assessment.....	59
Figure 23. Socratic Conversations History.....	60
Figure 24. User Profile Sections.....	61
Figure 25. User Settings.....	62
Figure 26. Microlesson History.....	63
Figure 27. Collaboration Session History.....	64
Figure 28. TITAN Platform Home Page.....	65
Figure 29. CT Assessment Section.....	65
Figure 30. CT Assessment Questionnaire.....	66
Figure 31. CT Questionnaire View from History.....	66
Figure 32. Microlesson Collections.....	67
Figure 33. Microlessons in a Collection.....	67
Figure 34. Microlesson View.....	68
Figure 35. Socratic Chatbot Section.....	69
Figure 36. New Conversation Creation.....	69
Figure 37. Socratic Conversation start.....	70
Figure 38. Socratic Conversation continuation.....	70
Figure 39. Socratic Message details.....	71
Figure 40. User Content in Socratic Conversation.....	71
Figure 41. Edit Chatbot message.....	72
Figure 42. Unlink messages.....	72
Figure 43. PIA initial page.....	73
Figure 44. PIA Session creation.....	73
Figure 45. PIA: selection or addition of social media credentials.....	74
Figure 46. PIA result.....	74
Figure 47. Collaboration initial page.....	75
Figure 48. Collaboration session creation.....	75

Figure 49. Collaboration session..... 76

Figure 50. Infrastructure Layers..... 77

Figure 51. Deployment via Docker Compose. .... 79

## LIST OF TABLES

<b>Table 1. Model Macro-F1 Score Performance across all classes and each class separately in parentheses on Disinformation Signal Detection Tasks.</b> .....	23
<b>Table 2. Hate Speech Classification results.</b> .....	24
<b>Table 3. Performance Comparison of Llama2 Base vs. Fine-Tuned Versions.</b> .....	25
<b>Table 4. Results from Additional Testing using a Subset of the Fine-Tuning Dataset.</b> .....	26
<b>Table 5. Confusion matrix of the initial PIA model.</b> .....	31
<b>Table 6. Use cases for User Management.</b> .....	46
<b>Table 7. Use cases for Critical Thinking Assessment Management.</b> .....	46
<b>Table 8. Socratic Conversation Management.</b> .....	47
<b>Table 9. Microlesson Management.</b> .....	47
<b>Table 10. Propagation Impact Assessment Management</b> .....	47
<b>Table 11. Collaboration Environment Management.</b> .....	48

## EXECUTIVE SUMMARY

This document reports on the activities and results of the design and implementation of the TITAN platform and its underlying services. More precisely, it encapsulates the comprehensive technical specification of the first TITAN platform release, including the minor releases delivered following the iterative development approach adopted for the project. These minor releases add new incremental features and integrate early feedback gathered during the Living Labs and the first round of pilots.

TITAN integrates socio-technical theory and user needs from WP2, refined through WP3's co-creation workshops and Living Labs. It employs an iterative development approach, utilizing feedback from WP3 and WP5 review sessions to guide development. The project implements an iterative requirement management process, promoting interdisciplinary collaboration to continuously align product features with target audience needs. This approach ensures effective translation of theoretical foundations, user requirements, and early feedback into technical specifications (see Section 2 and 3). Besides, to support agile principles such as iterative development lifecycle and continuous feedback loop, the TITAN implementation follows a continuous integration approach (see Section 10).

This iterative and collaborative process led to the evolution of the mock-ups' initial design (see Section 8.1), shaped by the initial user-journey and user-flows elaborated based on the requirements and partner insights, presented and revised during the co-creational workshops and then implemented in the first release of TITAN, and finally refined based on feedback from Living Labs and pilot use cases. The resulting TITAN Platform is a web-based application designed to implement the following key features:

**Evaluation of Critical Thinking Skills:** The platform assesses users' critical thinking skills, through interactive questionnaires that challenge users to think critically and analyze information (see Section 4.2).

**Detection of Disinformation Signals:** TITAN employs several AI-techniques to identify potential signals of disinformation in user-provided content. This involves natural language processing (NLP), machine learning, or text analysis to flag suspicious or misleading information (see Section 4.3).

**Proposing Microlessons on Disinformation:** The platform offers tailored microlessons to users based on the latest literature and research on disinformation and misinformation. These microlessons likely provide educational content aimed at improving users' understanding of how disinformation spreads and how to discern reliable information from false or misleading sources (see Section 4.6).

**User Engagement with a Socratic Conversational Agent:** TITAN features a conversational agent to engage users in discussions that follows the Socratic questioning method that promote critical thinking and deeper understanding of topics related to disinformation. The agent may ask probing questions to encourage users to reflect on their beliefs and reasoning processes (see Section 4.4).

**Evaluation of Propagation Impact on social media:** This component assesses the impact of content sharing on social media platforms, analyzing how information spreads and potentially influences public opinion. This evaluation could involve tracking metrics such as reach, engagement, and virality of shared content (see Section 4.7).

**User Engagement in Collaborative Argumentation Analysis:** TITAN provides a collaborative environment where users can engage in the analysis of arguments related to specific claims. This may involve structured debates, collaborative reasoning exercises, or argument mapping tools to visualize and evaluate the strength of different perspectives (see Section 4.8).

Overall, the TITAN Platform integrates various features and functionalities to enhance users' critical thinking skills, combat disinformation, and foster collaborative engagement in evaluating and understanding information.

This document provides comprehensive details on the TITAN components and the AI model that underpin various TITAN functionalities (Section 4). It also includes an in-depth explanation of the dataset used for training these models (Section 5) and the strategy for new dataset acquisition and curation.



The user-centric approach of the TITAN project, aiming to ensure a harmonious blend of functionality, usability, and impact, requires that the architectural choices and the structure of the platform be oriented to promote continuous development and seamless integration, while guaranteeing a strong focus on privacy (Section 3). In addition, given the rapid evolution of AI technologies and related challenges, TITAN undergoes an annual assessment to ensure compliance with legal and ethical requirements. To support the consortium in securing ethics approvals, especially for human participation and proper AI management, an External Ethics Advisory Board (EEAB) was established. Full compliance details will be included in the second version of the Report on Legal and Ethical Impact Assessment (D1.4).

To realize the TITAN Platform a microservice architecture has been adopted (Sections 0 and 7) and Graphical User Interfaces (GUIs) have been developed on top of it (Section 8).

Finally, the infrastructure and the continuous integration and delivery strategy for the TITAN Platform are explained (Sections 9 and 10).

## 1 INTRODUCTION

D4.1 is the first significant milestone in our mission to combat disinformation through innovative and educational means. This milestone builds upon the socio-technical theoretical background and user needs elaborated in WP2, shaped and refined through co-creational workshops and Living Labs sessions conducted in WP3, and through an iterative development approach aiming to facilitate a fast feedback loop through review sessions conducted in WP3 and WP5 to ensure that the development process is heading in the right direction.

At the heart of our system lies an innovative coaching chatbot, designed to employ the Socratic method. This innovative approach involves the chatbot engaging users with probing questions intricately linked to articles riddled with disinformation. Beyond mere exposure of fallacies, the method serves as an educational tool, nurturing users' critical thinking skills. This unique approach stimulates thoughtful analysis, empowering users to discern the veracity of information and cultivate a vigilant mindset, fostering a resilient mindset against disinformation.

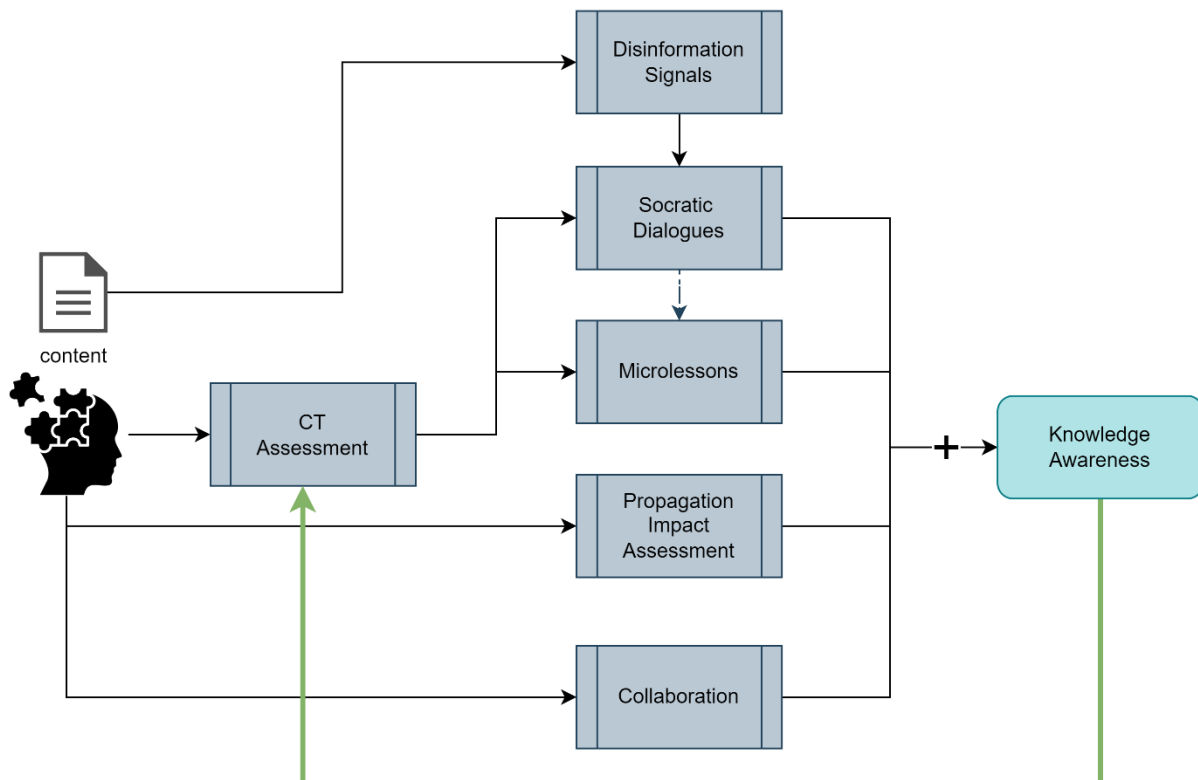
Chatbot dialogues are tailored to the user's critical thinking skills, which have been previously assessed using a psychometric model. To generate the proper dialogue, the chatbot works in synergy with several classifiers used to recognize possible disinformation signals. Insights gleaned from these classifiers provide a nuanced understanding of disinformation, enhancing the chatbot's ability to discern and engage in meaningful discussions. This integration elevates the platform beyond conventional approaches, ensuring a dynamic and adaptive response to the evolving landscape of disinformation.

Two additional components were conceived for the TITAN Platform to increase awareness and combat the spread of disinformation: the Propagation Impact Assessment (PIA), aiming to give an indication of "how far" the content at hand will spread if the user shares it; and the Collaboration Environment, allowing users to cooperate with each other on critical analysis of a variety of claims through structured argumentation.

Since all these features are AI-based, the TITAN Platform requires a set of benchmarking and training datasets to facilitate continuous improvement and adaptability. As users interact with the system and it evolves, these datasets become crucial in ensuring its increased effectiveness against disinformation.

### 1.1 General description of the framework components and their relations

The user requirements matured reviewing through an iterative requirement management process the work carried out by WP2 and WP3 delineate the main core components of the TITAN Platform and their relations (Figure 1). At the center, there is the user who provides as input some piece of information, the *content*, which in turn represents the principal focus for most of the functionalities offered by the system. The most outstanding functionality offered by TITAN is represented by the *Socratic Chatbot*, a virtual assistant able to hold a conversation by following a Socratic-like method to help the user dive into the news by a personalized approach tuned around the *Critical Thinking skills* of the user. Furthermore, the *Socratic Chatbot* is also aware about the presence of *disinformation signals* of specific types within the *content* and these signals will be considered when building dialogues during a conversation.



*Figure 1. TITAN Platform components.*

Other relevant functionalities of TITAN Platform, also focusing on the user content are: the Microlesson, the Propagation Impact Assessment (PIA), and the Collaboration Environment. Microlessons are represented by multimedia content enriched with metadata for searching and retrieval. They are proposed by experts and consumed by users. The Socratic Chatbot will be able to propose microlessons during a conversation. The *Propagation Impact Assessment* component uses the content of an article/social media post and the user’s social media profile information to predict “how far” will the content spread if the user shares it. The Collaboration Environment allows users to cooperate with each other on critical analysis of a variety of claims in structured argumentation.

The principal goal of the TITAN Platform is to increase the level of critical thinking of users thanks to the experience offered by the functionalities of the system.

In general, the TITAN Platform is built upon a heterogeneity of AI-based models and services that will be presented through a Graphical User Interface (GUI). The main components foreseen by the system can be summarized as follows:

- User Content. Any content used by users will be managed by this dedicated component (Sections 0, 7.3.2).
- Critical Thinking Assessment. This component is responsible to propose questionnaires to users in order to evaluate their level of critical thinking along different critical thinking dimensions, according to the Critical Thinking Assessment model developed in WP2 (Sections 2.2, 4.2, 6.2.2, 7.3.3).
- Disinformation Signal. Disinformation signals within a text will be detected by means of AI-based models and services (Sections 2.3, 4.3, 5.1).
- Microlessons. Collections of multimedia files will be organized by expert users and proposed to the final users as informative and educative material (Sections 2.2, 4.6, 6.2.4, 7.3.5).
- Socratic Chatbot. An AI-based conversational agent capable of conducting a Socratic-style dialogue to help the user engage in the verification process of the proposed content. The dialogue is personalised,

considering both the disinformation signals detected in the content and the user's critical thinking skills. (Sections 2.3, 4.4, 5.6, 6.2.3, 7.3.4).

- Propagation Impact Assessment. This component offers AI-based models and services to predict the potential spreading level of an article on social media network (Sections 2.4, 4.7, 6.2.5, 7.3.6).
- The Collaboration Environment. This component is based on computer assisted argumentation in framework of anonymous collaborative argument mapping. Users can view existing discussions and start new topics. The component offers the ability to discuss the merit of a variety of claims in structured graph conversations. Every claim can be addressed by premises, counterarguments, paraphrases or by providing sources. In this process new claims emerge and can be addressed. Claims connected with argumentative relationships can be reused in every context in which the claim is relevant, therefore distributed conversations create a bigger picture of network of evaluated claims by collective intelligence of the users (Sections 2.5, 4.8, 6.2.6, 7.3.7).
- Datasets collection and curation. Socratic conversations collected into the system will be used to fine tune the AI-based models employed to implement the Socratic Chatbot. Functionalities will be provided to view, correct and annotate Socratic conversations in order to produce better datasets along the project phases (Section 5).

## 1.2 Related Work Packages and Tasks

The design and implementation of the TITAN Platform was firstly based on the theoretic groundwork and the user needs provided by Work Packages 2 and 3, then iteratively reviewed and integrated with early feedback resulting from the Living Labs and first pilot use cases.

This deliverable presents the work carried out for the following tasks within Work Package 4:

- T4.1 - TITAN Services, Platform, and Architecture Specification: Sections 2, 3, 4, 5, 7, and 8.
- T4.2 – AI algorithms and techniques for Implementing the TITAN Trustworthy Services: Section 2, 4, 5.
- T4.3 - Co-creation of Training Data Sets: Section 6.
- T4.4 – Development of TITAN Services and Ecosystem Platform: Section 2, 3, 4, 5, 9, 10.
- T4.5 – Continuous System Integration, Testing, and QA: Section 11.

## 2 MAPPING OF USER REQUIREMENTS INTO THE TITAN PLATFORM COMPONENTS

TITAN builds upon the socio-technical theoretical background and user needs elaborated in WP2, shaped and refined through co-creational workshops and Living Labs sessions conducted in WP3, and through an iterative development approach aiming to facilitate a fast feedback loop through review sessions conducted in WP3 and WP5 to ensure that the development process is heading in the right direction. To properly and effectively transfer the theoretical foundation, the user needs, and early feedback into product features and technical specifications, TITAN applies an iterative requirement management process, fostering continuous interdisciplinary collaboration and ensuring that requirements reflect the actual needs of the target audience. The following subsections details the management process and report on the mapping of the reviewed requirements ready for implementation to the main components of the first release of TITAN.

### 2.1 Iterative Requirement Management

For an innovative research project like TITAN, gathering and refining requirements is not a one-time activity but a continuous process throughout the project lifecycle. Requirements are often not fully specified at the project's outset, frequently undergo changes, or are supplemented with new additions.

TITAN has adopted an Iterative Requirements Management approach, a methodology that prioritizes collaboration, user feedback, and flexibility in addressing evolving needs. Collaborative requirement analysis establishes a common understanding of the features that need to be developed, thus minimizing the role-based compartmentalization often observed in projects. By focusing on the iterative delivery of functional software increments, teams can evaluate and adapt system functionality as it progresses through development phases, ensuring customer satisfaction through regular communication and feedback cycles.

It involves a multidisciplinary team, including SSH (Social Sciences and Humanities), technical, and end-user partners. This team collaborates in periodic requirements reviews to analyze, categorize, prioritize and translate well defined user requirements into technical features for the TITAN backlog. This process involves also recognizing unclear or undefined requirements needing further clarification, thus reducing the risk of incomplete features being pushed into development, and the integration of early feedback, supported by a continuous integration and delivery approaches.

Iterative Requirements Management activities started reviewing the results of the activities conducted in WP2 and reported in D2.1. WP2 played a crucial role in the TITAN project, serving as the focal point where theoretical knowledge, user-centered viewpoints, and technical requirements came together. The literature review, a pivotal component of WP2, delved into the vast landscape of disinformation, unearthing key trends, challenges, and mitigation strategies. This foundational exploration, coupled with an in-depth understanding of user needs, resulted in a huge collection of user and technical requirements to be reviewed and iteratively translated into tangible product features and technical specifications.

In the translation of requirements into technical specifications, we have methodically organized each requirement within distinct contexts, ensuring a thorough coverage of user needs. Our approach involves categorizing these requirements based on specific areas and subjecting them to a comprehensive review process.

The **contexts** identified for initiating the primary grouping layer include the following:

**Chatbot:** Our technical specifications involve the implementation of natural language processing, dialogue management, and personalized response algorithms. This ensures an intelligent and interactive chatbot experience.

**Disinformation Signals:** Advanced algorithms form the backbone of content analysis, allowing the identification and flagging of disinformation signals. This strengthens our capacity to counteract misleading information.

**Microlessons:** To support concise and targeted learning experiences, our framework includes features facilitating the delivery and tracking of microlessons.

**Collaboration:** Integrating collaborative tools, such as discussion forums and group project functionalities, our framework enhances knowledge sharing through interactive platforms.

**CT Assessment:** The framework integrates features designed to create and deliver CT assessment. This encompasses the incorporation of question banks, scoring mechanisms, and adaptive difficulty levels.

**Monitoring:** Comprehensive monitoring tools are embedded to capture user activities. This data analysis provides insights for continuous improvement and personalized learning experiences.

**User Content:** User-friendly interfaces are developed to support content sharing and collaboration. Robust management features ensure a dynamic and engaging learning ecosystem.

**User Profile/Preferences:** Customization options are incorporated for user profiles and preferences. This allows personalized content recommendations and adaptive learning paths.

**Incentives:** Integration of incentive mechanisms, such as rewards and recognition, encourages active user engagement.

Within the framework, each technical requirement is systematically classified under specific **areas**:

**Core:** These are fundamental functionalities considered essential for the framework's operation and success.

**Informative:** These elements provide valuable information to users, enhancing their overall learning experience.

**Notifications (to users):** Functionalities designed to alert users about relevant updates and changes within the framework.

**Feedback (from users):** Components dedicated to collecting and processing user feedback, ensuring continuous improvement.

**Legal and Ethical:** This area emphasizes the importance of compliance with legal and ethical standards in the development and deployment of the framework.

Furthermore, throughout our Iterative Requirement Management process, each requirement is assigned one of the following **status**:

**Rejected:** These are not accepted, and clear reasons are provided for the rejection.

**To Be Decided (TBD):** Further discussion and resolution are needed for these requirements.

**To Be Revisited (TBR):** Accepted but identified for improvement or potential merging with other requirements.

**Accepted:** These have been approved and planned for implementation.

**Implemented:** ready for review.

**Completed:** reviewed and approved.

Moreover, the requirements are categorized according to the release version of TITAN, distinguishing between the initial and subsequent releases.

Finally, each requirement is **prioritized** using the Moscow method <sup>1</sup>:

**Must Have:** These are critical for the framework's functionality and success.

**Should Have:** Important but not critical for the initial release.

**Could Have:** Desirable but not essential, may be considered for future releases.

This organized methodology ensures that our technical requirements are well-structured, prioritized, and aligned with the changing needs of our users, facilitating a seamless integration into the framework.

In the following sub-sections, we go into details for all the requirements listed in Table 18.1 (Annex 2) of D2.1 considering the described methodology and grouping them by contexts.

Finally, in the subsequent sections of this deliverable, we delve into the specific technologies employed to seamlessly integrate user requirements into the technical aspects.

### **Legal and Ethical requirements & External Ethic Advisory Board (EEAB) recommendations compliance**

Considering the rapid evolution of the AI technology landscape and consequently of the related criticalities, a full assessment of the TITAN implementation against legal and ethical requirements is carried out on an annual basis. In addition, to guide the consortium in obtaining ethics approvals for the participation of humans (safeguarding proper handling of personal data and managing AI issues properly), an External Ethics Advisory

---

<sup>1</sup> [https://www.agilebusiness.org/page/ProjectFramework\\_10\\_MoSCoWPrioritisation](https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation)

Board (EEAB) has been set up in TITAN. The iterative requirement management process is conducted considering the recommendations of the EEAB. The details of the TITAN implementation compliance with these requirements and recommendations will be fully reported in the second version of the Report on legal and ethical impact assessment (D1.4).

## 2.2 Critical Thinking Assessment

The evaluation of the critical thinking of users is specified by requirements CT-1, CT-2 and CT-3. Only user requirement CT-3 has been flagged for the First Release of TITAN, and it refers to a multiple-choice test to be proposed to users in order to evaluate their level of critical thinking.

## 2.3 Socratic Chatbot, Disinformation Signals, Microlessons & Incentives

Reviewing the requirements originally listed in Table 18.1 (Annex 2) of D2.1, several crucial requirements to enhance the user experience of the TITAN's Socratic chatbot and to prevent disinformation have been identified for the first release of TITAN to be used in the first pilot execution.

- For the first version of the TITAN Socratic Chatbot, ten requirements have been identified concerning the Socratic Chatbot, the Disinformation Signals, the Microlessons and the Incentives.
  - The requirements with IDs BB-6, BB-9, UC3-1, UC3-2, DS-1, DS-3 and DS-4 have been accepted for implementation, while the requirements with IDs BB-14, BB-15 and BB-16 are still under discussion.
  - The IDs BB-6, BB-15 and BB-16 have been categorized as "must-have" priorities for TITAN's first release version, while the IDs BB-9, UC3-1, UC3-2, DS-1, DS-3, DS-4 and BB-14 are considered requirements that the first release should have.
- For the next versions of the TITAN Socratic Chatbot, twenty-nine requirements have been identified concerning the Socratic Chatbot, the Disinformation Signals, the Microlessons and the Incentives.
  - The requirements with IDs BB-12, UC1-5 have been accepted for implementation, while the requirements with IDs UC1-2, UC1-4, UC1-9, UC1-10, UC1-12, UC1-13, UC2-1, UC2-6, UC3-7, UC3-8, UC3-9, CC-14, CC-15, CC-17, CC-18, SM-1, SM-2, SM-3, SM-4, DS-2, DS-5, MI-1, MI-3, MI-4, S-1, S-2, S-4 are to be discussed further.
- The requirements with respective IDs CC-14, CC-17 must be implemented, while the requirements BB-12, UC1-2, UC1-5, UC1-10, UC1-12, UC1-13, UC2-1, UC2-6, CC-15, SM-2, SM-3 should be implemented. Finally, the requirements UC1-4, UC1-9, UC3-7, UC3-8, UC3-9, CC-18, SM-1, SM-4, DS-2, DS-5, MI-1, MI-3, MI-4, S-1, S-2, S-4 could be implemented for the next releases.

## 2.4 Propagation impact assessment

The user requirements related to the propagation impact assessment component are listed in D2.1 Annex 2 (Table 18). Based the priority level of the requirement, the requirements can be listed as follows:

- PIA-1 and PIA-2/UC1-3 have been categorized as “must have” requirements.
- PIA-3 and PIA-4 were categorized as “should have” requirements
- PIA-5 was categorized as an optional feature (“could have”)

The requirements PIA-1, PIA-3 and PIA-4 were accepted for implementation and also PIA-2/UC1-3 which is related to uploading article content to the platform. While originally scheduled for the second release, an initial (non-optimal) implementation of PIA-1, PIA-2/UC1-3, PIA-3 and PIA-4 has been released during the first round of pilot to gather early qualitative feedback. Improved version is expected for the second release.



## 2.5 Collaboration Environment

The Collaboration Environment (CE) provides a unique space in the TITAN platform for structured group communication with other humans in the same language rather than AI-based chatbot. Following technical specifications (see 4.8) are related to the following reviewed requirements:

- CC-21 “The tool could include a dialogue between users.” - Users will be able to communicate by exchanging arguments on shared maps.
- UC1-5 “Can start a session (conversation), pause it, and continue later” CE will function like internet forum. - Users will be able to come back to their discussion. However, it can look different in time, because it can get bigger when other users will add new arguments to the map.
- UC1-11 “Users want to share something they have learned”. - Users will be able to formulate things they have learned by adding new arguments in the appropriate context.
- UC2-2 “As a user, I want the ability to collaborate (in real time) with other NGO experts on analyzing articles and signals”. CE will allow for collaboration in real time on analyzing any claims.
- UC2-8 “As a user, I want the ability to create and share content, such as articles or analyses, with the community.” Any argument map will be automatically shared with community members that speak the same language.
- UC3-3 “Dialogues should be presented in multiple languages” - It is envisioned that user will be able to choose different languages in which argument mapping will be conducted. Given that one of the features of this module is connecting arguments from different discussions, the language data will be stored to propose claims from the past in correct language. The module will reflect the number of languages used in the TITAN app.

Given that the focus of CE must be aligned with main goals of the TITAN app, it must enhance the critical analysis of the reasoning and good practices of verifying information as stated in following requirements:

- BB-14 “When critical thinking is leading the user to dismiss factual information, the user may need to be encouraged to build its argument on factual sources.” While it is not feasible to premise every argument on factual source users will be encouraged in input box to select if their argument is based on source or just their opinion. If they choose a source, it will be displayed as a special relation to the claim in argument map for all to see it.
- UC1-17 “As a journalist I can change manually the detected signals of disinformation & arguments, so that I can correct mistakes the system made” in context of argumentation if the journalist will have moderator access, they will be able to edit arguments, however it is envisioned that in general bad arguments should be addressed by good counter arguments by all users journalist or otherwise. The way module works this counter argument will be inherited in every context in which this bad argument is repeated. In this way even bad arguments can be educational.

Lastly CE must provide good user experience which can be assessed by gathering feedback about the tool, explain the format of collaboration clearly and respect the privacy of the user as well as the amount of input needed for satisfactory experience:

- UC2-9 “As a user, I want the ability to provide feedback on the tool's functionality and content.” First version of the feedback form was implemented. It remains to be decided if it remain separate form for the module, or be integrated with overall TITAN app.
- UC3-5” “The key features/ services of the app should be demonstrated through a video or animated tutorial” CE will be demonstrated through tutorial.
- CC-7 “The tool should only ask the user to provide the data necessary for the service.” CE requires only arguments as an input, no other data is needed.



## 3 TECHNICAL REQUIREMENTS

Section 2 of this document presents an overview of the iteratively reviewed user requirements originally delineated in document D2.1, which are *currently* being addressed by single components of the TITAN Platform. These requirements encompass both *core* and *specific area* for each primary element of the system. The user requirements, along with initial mock-ups (Section 8.1), clearly define the *overarching landscape* for the TITAN application as a web-based platform. Consequently, there arises a necessity to specify essential general technical requirements crucial for an optimal framework solution. As a result, detailed elaboration has been provided on general and non-functional requirements for the project, including aspects such as performance, scalability, modularity, usability, security, among others.

### 3.1 Common Design Concepts

#### 3.1.1 Privacy by Design

Privacy by Design (PbD) underscores the importance of incorporating robust privacy principles and measures into the initial design and development phases of systems, processes, products, and technologies. The objective is to seamlessly integrate privacy considerations into every step of the development lifecycle, thereby prioritizing privacy as an integral element of the design rather than an add-on.

The fundamental principles of Privacy by Design that must be taken into account include:

- **Proactive Approach:** Privacy by Design advocates for a proactive stance rather than a reactive one when it comes to privacy. It encourages organizations to anticipate and prevent privacy breaches before they occur rather than responding to them afterward.
- **Privacy as Default Setting:** Privacy by Design stresses the importance of privacy being the default setting. Systems, applications, or processes should prioritize user privacy automatically, offering maximum privacy protections without requiring users to take extra steps.
- **Full Functionality:** Privacy measures should not undermine the functionality or usability of products or services. Privacy by Design aims to strike a balance where privacy protections seamlessly coexist with intended functionality.
- **End-to-End Security:** This involves implementing robust security measures throughout the entire data lifecycle, from collection to storage, processing, and disposal. It ensures data integrity, confidentiality, and availability.
- **Visibility and Transparency:** Privacy by Design highlights the necessity of providing clear and easily understandable information regarding data collection, usage, and management. Transparency fosters trust and enables individuals to make informed decisions about their data.
- **Respect for User Privacy:** The concept advocates for respecting user privacy preferences and individual autonomy. Users should have control over their personal data and its utilization.

Implementing Privacy by Design necessitates collaboration among developers, designers, privacy professionals, legal experts, and business leaders. It entails a shift in mindset, viewing privacy as an integral consideration rather than an afterthought, across all development and operational phases.

#### 3.1.2 Microservice Architecture

Microservices are a software architectural approach where an application is structured as a collection of loosely coupled, independently deployable services. Each service, or microservice, is designed to perform a specific business function and can be developed, deployed, and managed separately. Microservices communicate with each other through well-defined APIs, typically over HTTP or messaging protocols.

Microservices are commonly used for backend applications due to their ability to break down complex systems into smaller, manageable services. However, they can also be utilized in front end applications, especially in

scenarios where the frontend interacts with various backend services or APIs. In such cases, frontend components may be designed to work as micro-frontends, allowing for independent development and deployment of different frontend modules.

Microservices break down systems into smaller, independent units, offering numerous benefits for distribution and maintainability:

- **Scalability:** Microservices allow individual components to scale independently, optimizing resource usage and responsiveness.
- **Modularity and Flexibility:** Each microservice performs a specific function, facilitating modification, updates, or replacements without affecting the entire system, leading to faster development cycles and easier adaptation to new requirements.
- **Distribution and Deployment:** Microservices can be deployed independently across various environments, such as on-premises servers, cloud platforms, edge devices, or IoT devices, enhancing distribution flexibility.
- **Fault Isolation and Resilience:** Microservices isolate faults, ensuring that a failure in one service does not disrupt the entire system, thereby improving system resilience and availability.
- **Technology Diversity:** Different parts of the AI application can be developed using various technologies, languages, or frameworks, optimizing each service for its specific function and enabling the use of specialized tools or libraries.
- **Parallel Development:** Teams can independently work on different microservices, accelerating development and enabling parallel work on various AI application components.
- **Integration and Interoperability:** Microservices communicate through well-defined APIs, facilitating integration with other services or external systems and promoting interoperability and seamless interaction between different components.
- **Resource Efficiency:** Microservices can be optimized for specific workloads, enhancing resource utilization and reducing overhead compared to a monolithic architecture.

## 3.2 General Requirements

The subsequent sections outline additional requirements that address key aspects, considering the technology's target audience, which includes the general public. These requirements comply with legal and ethical standards, as well as with the recommendations of the External Ethics Advisory Board (EEAB). Details of these standards and recommendations are outlined in project deliverables D1.3 and D1.4. Each requirement is numbered for easy reference.

### 3.2.1 Reliability

[R-RELIABILITY-01] High Mean Time Between Failures (MTBF): The developer must aim for a significantly high MTBF for the application or system, ensuring prolonged operation without failure.

[R-RELIABILITY-02] Robust Error Handling: Comprehensive error handling mechanisms must be implemented to gracefully manage exceptions, prevent system crashes, and maintain stability across various conditions.

[R-RELIABILITY-03] Thorough Testing: Rigorous testing, including stress testing, fault injection, and failure mode analysis, should be conducted to identify and rectify potential failure points and weaknesses.

[R-RELIABILITY-04] Resilience and Redundancy: Sufficient built-in redundancy, failover mechanisms, and resilience features must be implemented to minimize downtime and ensure uninterrupted operation, even in the face of failures.

### 3.2.2 Availability

[R-AVAILABILITY-01] High Uptime: The developer should target a high level of uptime for the application or system, typically exceeding 99% availability, ensuring accessibility and operation for users.

[R-AVAILABILITY-02] Quick Recovery: In the event of a failure, mechanisms should be in place to enable the application or system to quickly recover and restore services without significant impact on users.

[R-AVAILABILITY-03] Load Balancing and Scalability: Considering availability requirements, the developer should implement load balancing techniques and scalability to manage varying workloads and prevent resource overload.

### 3.2.3 Monitoring

[R-MONITORING-01] Real-Time Monitoring: The developer should deploy comprehensive monitoring tools and systems to track performance metrics, detect issues proactively, and provide insights for optimizing reliability and availability.

### 3.2.4 Data and System

[R-DATA/SYSTEM-01] Data Integrity: Robust data recovery mechanisms should be implemented to ensure data integrity and prevent loss in case of failures.

[R-DATA/SYSTEM-02] Regular Backups: The maintenance organization should establish procedures for scheduled and redundant backups of critical data and configurations, facilitating quick restoration in case of failure.

[R-DATA/SYSTEM-03] Modularity through APIs and Microservices: The TITAN Platform should adopt a design emphasizing modularity and flexibility, achieved through a microservices architecture.

[R-DATA/SYSTEM-04] Scalability as a Core Requirement: The TITAN Platform should prioritize scalability, both horizontally and vertically, to accommodate increasing data volumes and user demand.

[R-DATA/SYSTEM-05] Tailored Data Storage Solutions: The TITAN Platform should select appropriate data storage technologies based on the nature of the data, including relational databases (e.g., MySQL, PostgreSQL) for structured data, NoSQL databases (e.g., MongoDB, Cassandra) for semi-structured or unstructured data, and data lakes for vast amounts of raw data.

[R-DATA/SYSTEM-04] Robust Data Integration Mechanisms: TITAN Platform should implement strong data integration methods, leveraging APIs, connectors, and Extract Transform Load (ETL) processes for efficient data collection, cleansing, and standardization into a consistent format.

[R-DATA/SYSTEM-05] Stringent Data Security Measures: The TITAN Platform must incorporate comprehensive security measures, including encryption, access controls, and regular security audits, to protect sensitive data against breaches.

[R-DATA/SYSTEM-06] Compliance and Governance Standards: The TITAN Platform should adhere to robust data governance practices to uphold data integrity, quality, and privacy.

[R-DATA/SYSTEM-07] Comprehensive Monitoring and Management Tools: Advanced monitoring tools and systems should be integrated into the TITAN Platform to monitor performance, resource utilization, and data quality. Automated alerts and routine maintenance should complement these tools for timely issue resolution.

[R-DATA/SYSTEM-08] Fault Tolerance and Reliability Features: The TITAN Platform should incorporate fault-tolerant mechanisms to ensure continuous operation, even during hardware failures or system issues. Strategies such as redundancy, replication, and backups should be implemented to enhance fault tolerance and reliability.

### 3.2.5 Security

[R-SECURITY-01] Resilience to Security Threats: Robust security measures must be implemented to safeguard against potential security threats and vulnerabilities that could impact availability or reliability.

[R-SECURITY-02] Secure Communication Protocols: The TITAN Platform must utilize secure communication protocols like TLS to safeguard data transmission, preventing eavesdropping or interception.

[R-SECURITY-03] Encryption: Robust encryption techniques must be employed across the TITAN Platform for data at rest and in transit, utilizing strong encryption algorithms to protect sensitive information.

[R-SECURITY-04] Secure User Authentication: Preferably, the TITAN Platform should incorporate multi-factor authentication (MFA) for accessing critical systems, adding an extra layer of security beyond passwords.

[R-SECURITY-05] Access Controls: Strict access controls and authentication mechanisms, including role-based access control (RBAC), should be implemented to restrict access to sensitive data based on users' roles and permissions.

[R-SECURITY-06] Data Segmentation: Sensitive data within repositories should be segmented, limiting access based on the principle of least privilege to minimize exposure and mitigate the impact of breaches.

[R-SECURITY-07] Regular Audits and Monitoring: Continuous monitoring and regular audits should be established to track access patterns, detect anomalies, and ensure compliance with security policies. Logging and monitoring tools should be implemented to track user activities.

[R-SECURITY-08] Regular Maintenance: Scheduled maintenance routines, updates, and patches should be performed on the TITAN Platform to maintain stability and security without disrupting service.

[R-SECURITY-09] Data Backups and Disaster Recovery: Robust backup strategies and disaster recovery plans must be implemented to ensure data availability and integrity in the event of unforeseen events or security incidents.

[R-SECURITY-10] Compliance with Regulations: The TITAN Platform must comply with international data protection regulations such as GDPR and local data protection laws, fulfilling requirements for data storage, transfer, and processing in different regions.

[R-SECURITY-11] Legal Agreements and Contracts: Clear legal agreements and contracts should be established for the TITAN Platform, defining security responsibilities, liabilities, and data protection measures when dealing with international partners.

### 3.2.6 Auditability

[R-AUDITABILITY-01] Data Collection and Preprocessing: Details about datasets used for training and testing the AI model must be documented, including data sources, preprocessing steps (cleaning, normalization), transformations applied, and data augmentation techniques.

[R-AUDITABILITY-02] Model Architecture and Training: The architecture of the AI model, algorithms used, hyperparameters, and optimization techniques during training should be recorded. Document training specifics like epochs, batch sizes, learning rates, and any fine-tuning steps.

[R-AUDITABILITY-03] Validation and Evaluation: Document the model validation process, including performance metrics like accuracy, precision, recall, and F1-score. Record evaluation results on validation and test datasets.

[R-AUDITABILITY-04] Model Versioning and Deployment: Keep track of model versions, noting updates, improvements, or changes made. Document deployment environment and configurations.

[R-AUDITABILITY-05] Monitoring and Performance in Production: Create logs capturing the AI model's performance in production. Monitor metrics like inference time, accuracy, error rates, and feedback loops for continuous improvement.

[R-AUDITABILITY-06] Decisions and Explanations: Record decisions made during development, including algorithm choices, feature selection, and ethical considerations. Provide explanations for these decisions.

[R-AUDITABILITY-07] Data Handling and Privacy: Document measures taken to ensure data privacy, such as anonymization, encryption, access controls, and compliance with regulations like GDPR.

[R-AUDITABILITY-08] Error Analysis and Remediation: Maintain records of model errors, biases, or unexpected behaviour observed during testing or production. Document steps taken to address these issues and improve performance.

[R-AUDITABILITY-09] Security Measures: Detail security protocols implemented to protect the AI system from vulnerabilities, including encryption methods, access controls, and regular security audits.

[R-AUDITABILITY-10] Compliance and Regulatory Considerations: Ensure the AI application complies with relevant laws, regulations, and ethical guidelines. Document steps taken to address compliance requirements.

### 3.2.7 Deployability

[R-DEPLOYABILITY-01] Modular Design: Create modular and reusable components within the application, employing microservices or well-defined APIs for seamless integration with other systems.

[R-DEPLOYABILITY-02] Scalability: Design the AI application to scale horizontally and vertically, using scalable infrastructure, distributed computing frameworks, and containerization (e.g., Docker, Kubernetes) for efficient scaling.

[R-DEPLOYABILITY-04] Documentation and Tutorials: Provide comprehensive documentation, tutorials, and guides for easy onboarding, setup, and usage of the AI application, simplifying deployment for users and developers.

[R-DEPLOYABILITY-05] Monitoring and Logging: Incorporate robust monitoring and logging mechanisms to track the application's performance, resource utilization, and potential issues in real-time, aiding in prompt issue identification and resolution.

[R-DEPLOYABILITY-08] Feedback Loops and Iterative Improvement: Establish mechanisms for collecting user feedback and performance data post-deployment, utilizing this information to iterate and improve the application, addressing user needs and issues.

[R-DEPLOYABILITY-10] Compliance and Governance: Ensure compliance with relevant regulations and industry standards, including data protection laws and ethical guidelines, for wider adoption.

[R-DEPLOYABILITY-11] User-Friendly Interfaces: Develop intuitive user interfaces and experiences to facilitate easy interaction and adoption of the application, enhancing deployability and user satisfaction.

## 4 TITAN PLATFORM COMPONENTS

The following subsections describe the architectural components of the first major release of the TITAN Platform and subsequent minor releases, which were used during the Living Labs and the first round of pilots to gather early and frequent feedback.

### 4.1 User Content

User content serves as the primary input for TITAN functionalities, whether it's an article sourced from the web or free text. The Socratic Chatbot, the PIA, and the Collaboration environment rely on textual content provided by users to execute their tasks effectively. For instance, the Socratic Chatbot assesses both the substance of the content and any indicators of disinformation present within it.

The User Content component facilitates the storage and analysis of user-provided data. It communicates with other components to accomplish the desired functionalities within the system.

### 4.2 Critical Thinking Assessment

The Critical Thinking Assessment component is designated to implement the questionnaires needed to evaluate the critical thinking skills of the user, as indicated in deliverable D2.2.

### 4.3 Disinformation Signal Detection

In this section, we present the techniques used to develop various disinformation signal detection services, focusing on hate speech, offensive language, clickbait, and logical fallacies. A variety of Transformer-based language models and large language models (LLMs) were employed for binary and multi-class text classification tasks.

#### 4.3.1 Hate Speech, Offensive Language & Clickbait Binary Classification Models

The process of creating models for detecting disinformation signals involved collecting and combining datasets for hate speech, offensive language and clickbait in English. The next step involved fine-tuning several pre-trained language Transformer models taken from the Hugging Face platform, including XLM\_RoBERTa (Conneau, et al., 2020), RoBERTa (Liu et al., 2019), and mDeBERTa (He et al., 2023) for text classification.

For detecting hate speech and offensive language, two binary text classification models were created to differentiate between hate speech and offensive language. The binary classification models were designed to detect the presence or not of hate speech or offensive language in a text. Binary classification models were developed for clickbait detection as well. In this case, the models were tasked to detect whether a text contains clickbait or not.

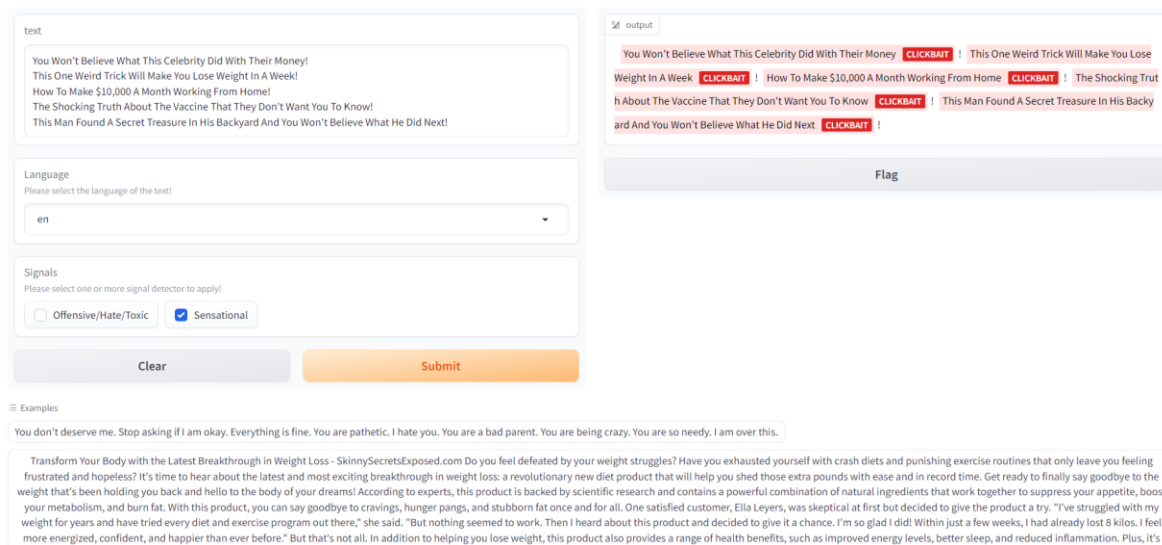
The models' performance was assessed using various metrics, including Macro-F1 score. The best-performing models, which are presented in Table 1 below, were then retained and integrated into the TITAN API for disinformation signal detection services, which enables users to detect hate speech and clickbait in texts. The disinformation signals API is available here: <https://titan.iit.demokritos.gr/titan/docs>. A demo to test the detection of the available disinformation signals using the "Gradio" platform was also set up and is available here: [https://titan.iit.demokritos.gr/demo/disinformation\\_signals/](https://titan.iit.demokritos.gr/demo/disinformation_signals/)

Task	Model	Macro-F1 Score
Hate Speech Binary Classification Text ( <i>HATEFUL/NOT</i> )	mDeBERTa-V3-Base2	88.49 (87.88 <i>HATE</i> , 89.09 <i>NOT</i> )
Offensive Language Binary Text Classification ( <i>OFFENSIVE/NOT</i> )	XLM_RoBERTa-Large	89.60 (92.57 <i>OFFENSIVE</i> , 86.63 <i>NOT</i> )



Clickbait Binary Text Classification (CLICKBAIT/NOT)	XLM_RoBERTa-Large	97.83 (97.96 CLICKBAIT, 97.69 NOT)
---	-------------------	---------------------------------------

**Table 1. Model Macro-F1 Score Performance across all classes and each class separately in parentheses on Disinformation Signal Detection Tasks.**



**Figure 2. The Gradio Demo Platform.**

#### 4.3.2 Multi-Label Hate Speech Detection model

The extensive popularity of the Internet and social media has empowered the fast and anonymous spread of Hate Speech content on platforms such as Twitter and Facebook. Due to the existing legislation against hateful language combined with the large amount of data produced in these platforms, arouses the need for tools that can detect Hate Speech (HS). Due to the large amount of data transmitted through these platforms, assigning this job to people is not an option. A common alternative is to rely on user reports to review only the reported posts and comments. This is also inefficient, since it relies on the users' subjectivity and trustworthiness, as well as their ability to thoroughly track and flag such content. Thus, the development of automated tools to detect Hate Speech content is deemed necessary.

A major difficulty and an important first step is to provide a clear and comprehensive definition of Hate Speech. It is a crucial process especially during the manual compilation of Hate Speech detection datasets, where human annotators are involved, since HS detection is a highly subjective process and depends on the background of the annotator such as education, culture, race etc. therefore, a concise definition is required to remove any personal bias in the annotation process.

The term Hate speech covers all forms of expressions that that spread, incite, promote, or justify racial hatred, xenophobia, antisemitism, or other forms of hatred based on intolerance. In addition, it can be insulting, degrading, defaming, negatively stereotyping or inciting hatred, discrimination, or violence against people in virtue of their race, ethnicity, nationality, religion, sexual orientation, disability, gender identity". Hate Speech can be also expressed by statements promoting superiority of one group of people against another, or by expressing stereotypes against a group of people.

To achieve this objective, we have selected to finetune a pretrained *distilroberta-base* model. The model is finetuned in a combined dataset which is based on 3 separate datasets in the context of hate speech, namely the CONAN (COunter NArratives through Nichesourcing) and the HKUST-MLMA (Multilingual and Multi-Aspect Hate Speech Analysis) dataset and the *'tweets\_hate\_speech\_detection'* dataset which is provided from datasets library.

We focus on user-generated texts from social media platforms, specifically Twitter posts. The types of interest refer to content that expresses stereotypes, generalizations and/or chauvinistic views that are examples of:

- **Non-Hateful**, referring to Non hateful content of a text
- **Racism**, i.e. related to the race of an individual or a group.
- **Sexism**, related to expressions of misogyny / misandry.
- **Sexual orientation**, connected to sexual orientation and gender identity.
- **Religious**, referring to the religion of an individual or a group.

The model's performance was assessed using various metrics, including Macro-F1 score. We also created a binary model based on a pretrained distilRoBERTa-base model in the same data by differentiate the labels, to be binary (HATE/NOT).

Task	Model	Macro-F1 Score
Hate Speech Binary Classification (HATE/NOT)	distilRoBERTa -Base	0.92
Hate Speech Multi-Label Classification (None, Racism, Sexism, Sexual orientation, Religious)	distilRoBERTa -Base	0.89

*Table 2. Hate Speech Classification results.*

### 4.3.3 Logical Fallacy Detection with Large Language Models

The natural language processing (NLP) capabilities of Large Language Models (LLMs) enable them to understand and interpret complex language structures, making them particularly effective in identifying patterns or reasoning, including logical fallacies and misleading contents. Through the examination of argumentative structure and content, these models contribute significantly to the identification of logical inconsistencies. This functionality assists in assessing information quality, enhancing the capacity to differentiate between logically sound arguments and those that are not.

To achieve this objective, we have meticulously selected the LLM that offers the optimal trade-off between performance, processing speed, and computational power. These models have been chosen based on their proven efficacy in accurately generating text under certain conditions. This careful selection ensures that our approach delivers high-quality results and remains feasible and sustainable regarding computational resources.

In order to effectively detect logical fallacies, a Large Language Model must be specifically trained for this task. To this end, we selected Llama2 with 13 billion parameters (meta-llama/Llama-2-13b) for its high potential in understanding and processing complex language patterns. We then undertook a comprehensive fine-tuning process, tailoring the model to specialise in identifying logical fallacies. This fine-tuning involved feeding the model diverse texts containing various logical fallacies and expert annotations to enhance its ability to recognise and analyse such fallacies accurately.

The impact of this fine-tuning process was significant. We observed a marked improvement in the model's performance compared to its base version. Post-fine-tuning, Llama2 demonstrated an increased accuracy in identifying logical inconsistencies and fallacies. This advancement highlights the importance of specialised training in augmenting the capabilities of Large Language Models for specific, nuanced tasks such as logical fallacy detection. As we delved into the fine-tuning process, we faced the challenge of limited availability of specialised datasets in the literature, particularly those containing news articles with annotated logical fallacies. To address this, we created a dataset called 'Augmented Logical Fallacy Dataset,' which will be described in detail in the following section. This dataset has been split into a rate of 90% for fine-tuning Llama2 and 10% to validate the process. In addition to enhancing the model's ability to detect logical fallacies, fine-tuning was crucial in structuring the model's responses. This structuring was necessary for efficient data extraction and analysis. We employed a JSON format for the model's output to facilitate this. This format



allowed us to systematically extract information from the model’s responses, particularly the identified logical fallacies and their specific characteristics. By structuring the output in a JSON format, we could pinpoint the exact location of fallacies within the text, categorise different types, and gather additional contextual information that the model provided about each identified fallacy.

In evaluating our model’s performance, we employed a set of established metrics: ROUGE, BLEU, METEOR, and BERT-Score. These metrics provided a multifaceted assessment, allowing us to gauge the model’s proficiency in accurately identifying and analysing logical fallacies. ROUGE and BLEU offered insights into the overlap between the model’s output and reference texts, METEOR provided an understanding of alignment with the reference segment of texts, and BERT-Score evaluated the semantic similarity between the model’s output and the reference texts. By utilising this diverse set of metrics, we ensured a comprehensive and robust evaluation of the model’s performance post-fine-tuning.

Our experiment demonstrated the effectiveness of fine-tuning Large Language Models for specific tasks, such as logical fallacy detection. We compared the performance of the base version of Llama2 with its fine-tuned counterpart to assess the impact of our training process. The comparison was performed using the “Disinformation Signal Dataset,” which revealed notable improvements in the model’s ability to identify logical fallacies post-fine-tuning accurately. This was evidenced by the increased scores across various metrics, indicating a higher accuracy and better alignment with reference texts.

Below is a table summarising the performance of the base and fine-tuned versions of Llama2 across different metrics:

Metric	Llama2 Base	Llama2 Fine-Tuned
BLEU	0.08	0.13
ROUGE	0.12	0.17
METEOR	0.11	0.26
BERT-SCORE	0.85	0.86

*Table 3. Performance Comparison of Llama2 Base vs. Fine-Tuned Versions.*

The results show a clear improvement in all metrics for the fine-tuned version of Llama2. The BLEU and ROUGE scores, which focus on the overlap with reference texts, increased, suggesting a more precise detection of fallacies. The METEOR score significantly improved, which evaluates alignment with reference translations, indicating a better understanding of the fallacies' context.

These results underscore the effectiveness of our approach in enhancing the Llama2 model's capability to identify logical fallacies, highlighting the importance of both specialised training and prompt engineering in achieving high performance in specific tasks.

To further validate the model's performance, we conducted another test using 10% of the dataset for fine-tuning the model. This additional test provided a more focused evaluation of the model's capabilities. The results from this test are summarised in the table below:

Metric	Llama2 Fine-Tuned
BLEU	0.73
ROUGE	0.76
METEOR	0.74
BERT-SCORE	0.95

*Table 4. Results from Additional Testing using a Subset of the Fine-Tuning Dataset.*

These results from the subset test are significantly higher than those from the initial evaluation, indicating a robust understanding and detection of logical fallacies by the fine-tuned Llama2 model. The substantial score increase across all metrics highlights the model's improved performance and accuracy in identifying logical inconsistencies and fallacies within texts.

#### 4.4 Socratic Chatbot

The first version of the Socratic Chatbot is based on a rule-based conversational model to ensure a structured and predictable conversational flow. This approach is crucial for gathering insights on usability, impact, acceptance, and attitude during the initial stages of Living Labs and the first Pilot iteration. The chatbot relies on 25 meticulously crafted flowcharts from handwritten Socratic dialogues crafted by a multi-disciplinary team composed of Social Sciences and Humanities (SSH), technical and end-user partners. These flowcharts guide users through the Socratic method steps consistently, tailored to different disinformation signals. The flowcharts depict the order and sequence of questions that can be asked for each of the Socratic method steps outlined in D2.1 (8.1.1), depending on the step and the disinformation signal. The flow charts for steps 1 (Clarification), 4 (Alternative Viewpoints), 5 (Implications and Consequences) and 6 (Challenging the Question) are common for all disinformation signals. The flow charts for steps 2 (Challenging Assumptions) and 3 (Evidence and Reasoning) vary depending on the signal.

The rule-based method allows the chatbot to customize interactions based on the selected news article, the user's critical thinking assessment, and the specific disinformation signal while maintaining a controlled dialogue flow. The predictability was crucial for the initial deployment phase and provided a stable foundation for future enhancements and iterations. The manually crafted Socratic dialogues were used as training data to develop the rule-based Socratic chatbot used for the first iteration. Thus, the current version of the chatbot can lead conversations using the Socratic dialogues' data. The chatbot can discuss the following disinformation signals: *hate speech, emotional language, slippery slope, cancelling hypothesis, unverified sources, implicit assumption, bandwagon, conspiracy theory, trolling, and discredit*. Each signal is encountered in specific utterances of the model, each taking place in a specific Socratic dialogue step.

The rule-based approach has been implemented as a Dialogue-as-a-Service (DaaS) using the Rasa<sup>2</sup> platform. It follows a traditional architecture, as presented in Figure 3. Rasa implements dialogues through a knowledge base organized in the form of "stories", which are sample dialogues that include state, intents, and actions. These stories are used by Rasa to learn dialogue policies that map state to actions through machine learning. Additionally, machine learning is employed to analyse user input to detect entities and classify the user input into predefined intents.

---

<sup>2</sup> Rasa Conversational Platform: <https://rasa.com/>.

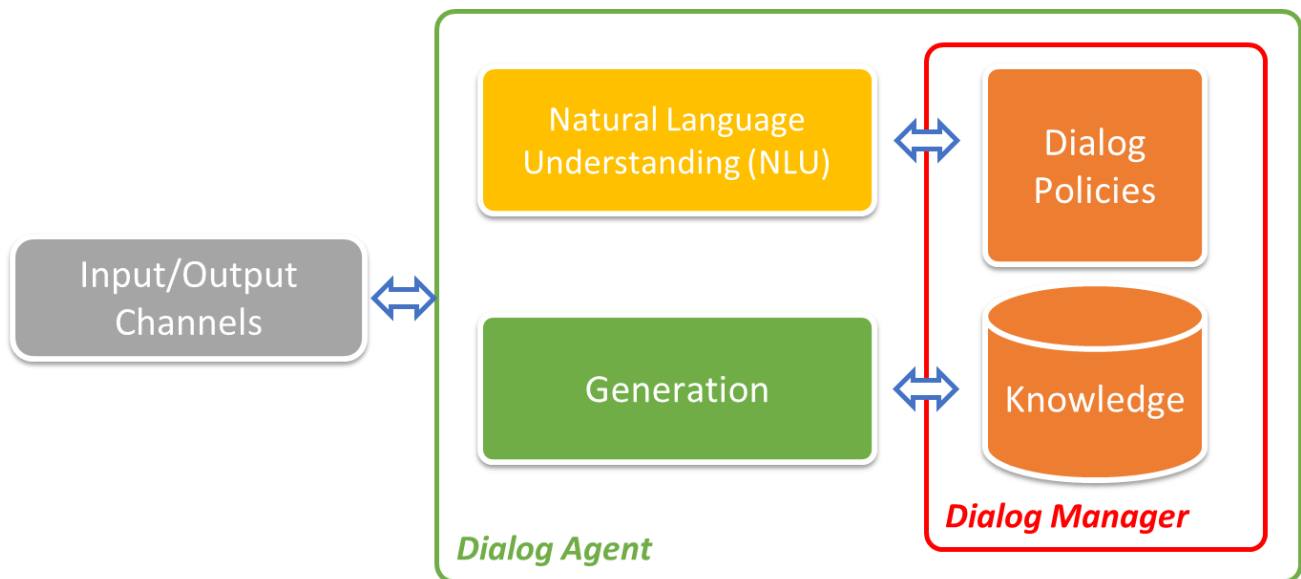


Figure 3. Rule-based architecture of Rasa Platform taken from RASA documentation.

The TITAN implementation analyses the disinformative content the user provides, such as a news item, to identify disinformation signals outside the Rasa platform. The analysis results are then used to initialize the dialogue state of Rasa. Subsequent decisions are made based on a combination of Rasa's learned policies and custom actions, influencing policies through custom "slots" used by Rasa to maintain the dialogue state. In the example provided in Figure 4, the flow of a single user input is depicted, showcasing how the input is classified into an intent and how an action is decided based on the current state of the policy, using knowledge extracted from sample dialogues expressed as a story, a sequence of intents, slots, and actions.

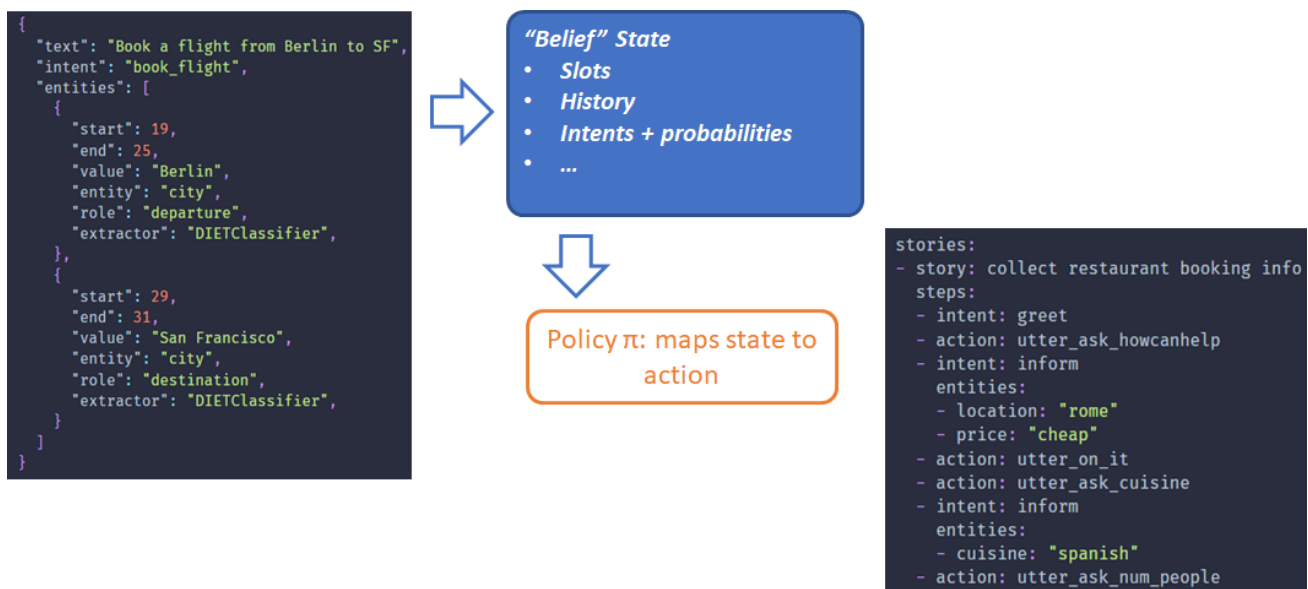
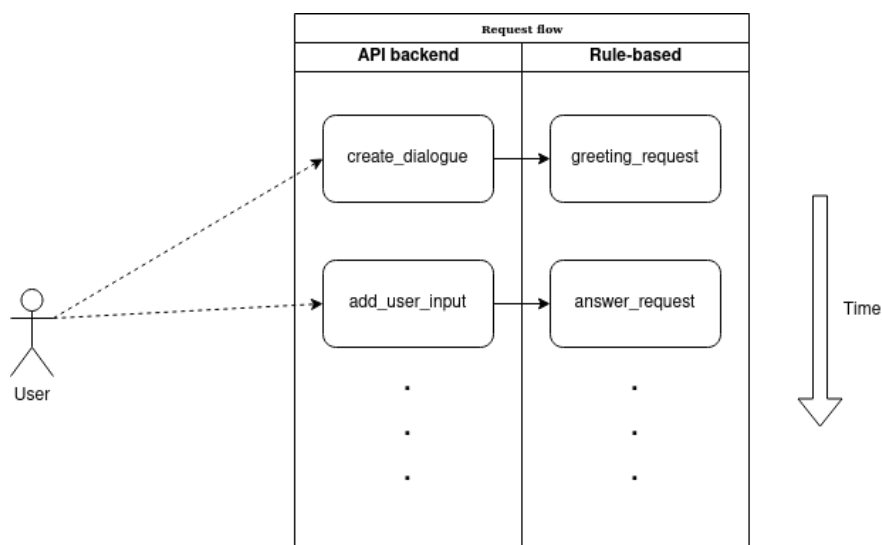


Figure 4. User input example in Rasa. Example taken from RASA documentation.

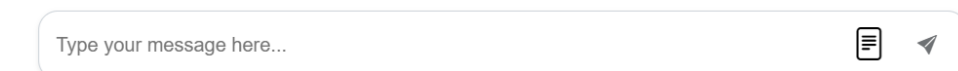
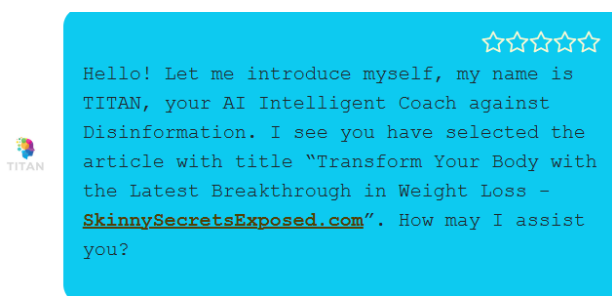
The system uses requests from the main API backend to handle the Socratic dialogue. It then stores the dialogue instance and context data (such as critical thinking assessment, selected article, and current Socratic step) in memory. The system begins by greeting the user and initiates a conversation based on the analysis of the context data. The rule-based system responds to each answer provided by the user with additional questions, until all steps of the selected dialogue setup are completed. Alternatively, the user can choose to end the dialogue process at any time. The user can also modify the dialogue by changing the persisted text or deleting a dialogue.



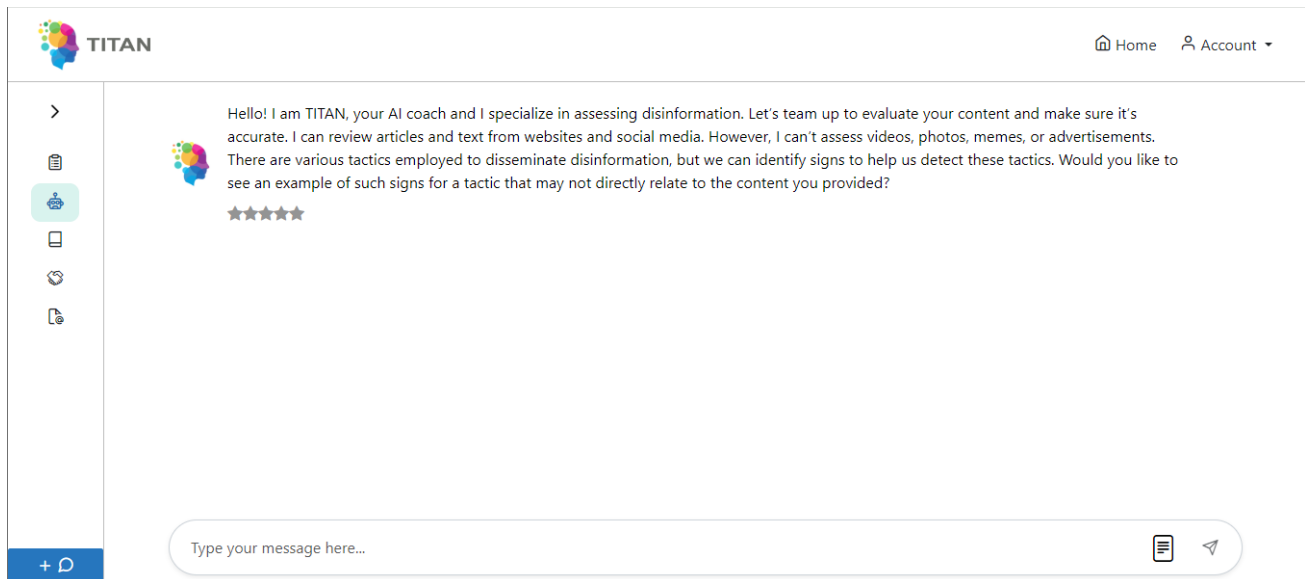
*Figure 5. Outline of the system communication between the API backend and the rule-based part.*

A user can chat with the Socratic Chatbot through this platform: <https://titan-coach-platform.eu/>

Figure 6 shows the first major release implementation of the TITAN Socratic Chatbot UI, while Figure 7 shows its latest minor release implementation integrating some of the early feedback gathered during the living labs and pilots.



*Figure 6. First version of the TITAN Socratic Chatbot UI.*



*Figure 7. Latest version of the Socratic Chatbot UI*

#### 4.5 Paraphraser

The Llama 2 chat model, which boasts 7 billion parameters in its quantized form due to memory limitations ('TheBloke/Llama-2-7b-Chat-GPTQ'), was utilized to create a highly effective contextual response paraphraser. After conducting many experiments, we were able to develop a sophisticated system prompt that enables successful paraphrasing of chatbot questions while preserving the context. The user instruction was also provided with a variable that takes as input the chatbot response: "Paraphrase the following question. Question to paraphrase: {question}".

The key objective of developing the Llama 2 paraphraser was to rephrase the chatbot responses produced by our rule-based system. The paraphraser model was designed to create a diverse range of chatbot responses, which we subsequently added to the rule-based training set responses, thus enhancing the training set. This approach was taken to ensure that the TITAN chatbot could provide a variety of helpful and accurate responses to users, hence improving the overall user experience.

#### 4.6 Microlessons

Microlessons consist of multimedia files pertinent to the context of disinformation. In the current phase of the project, users can access microlessons from a dedicated section where each document is available for direct viewing on the platform or for download. Additionally, the Socratic Chatbot has the capability to suggest proper microlessons to users as needed.

#### 4.7 Propagation impact assessment

PIA component has been implemented as a stand-alone module and it is compiled to a Docker container. To integrate the trained machine learning model with the rest of the TITAN system, HTTP based interfaces have been developed for the purpose, using JSON formatted data structures as HTTP POST payloads. When the TITAN system makes a request for a PIA estimate, it provides the content of the post/article, username of the user's Mastodon profile and possible metadata as JSON payload. The PIA component then returns the PIA estimate in JSON format.

An illustration of the architecture is shown in Figure 8.

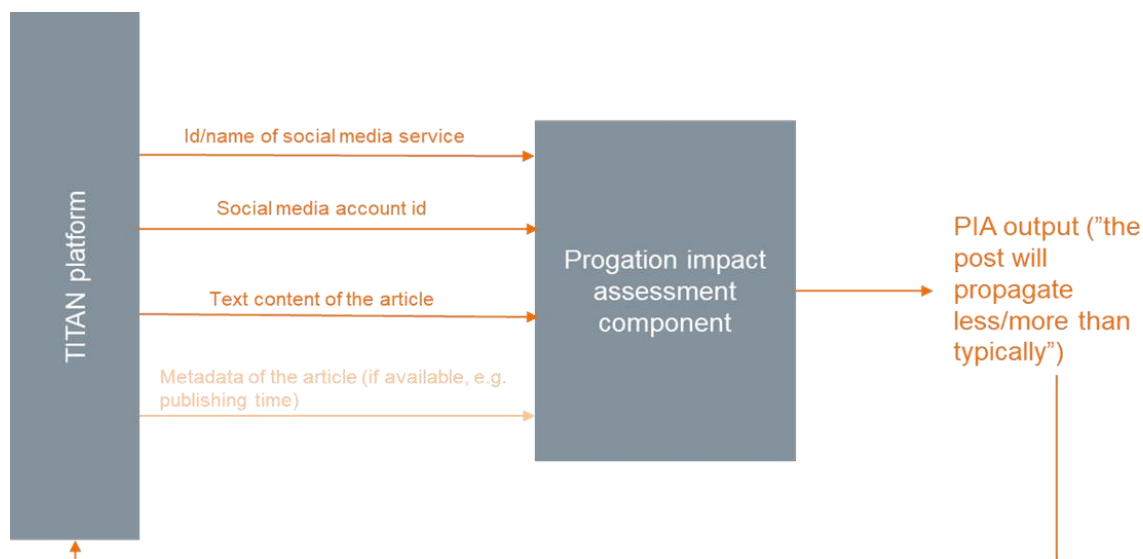


Figure 8. PIA component and its connections to TITAN platform.

Example JSON structure that is taken as an input by the PIA component:

```

{
  "username": "@username@mastodon.xyz",
  "content": "Fake vaccinations are spreading globally. #news",
  "publish_time": "2023-09-03T12:41:04.979Z",
  "number_of_media_attachments": 3
}
  
```

Example JSON output of the PIA component:

```

{"predicted_shares": "less_than_usual"}
  
```

#### 4.7.1 Propagation impact assessment models

The approach used in PIA is based on machine learning. A model is trained to estimate PIA with a collected dataset, after which the trained model is used as the core of the propagation impact assessment component. To create an input for the model, text content and social media profile information of the user are translated into features that present different aspects of the post and author of the post in numerical values. Examples of these are the polarity of the text content, number of characters and words in the text and number of followers of the user. The machine learning model then gives the estimate by utilizing these features and approximating the relation between these features and the propagation impact.

Experiments were done with two different PIA metrics. Initial experiments indicated that it is challenging to approach PIA as a regression problem (i.e. predicting the exact number of shares a post will have). Consequently, PIA was also approached as a classification problem, i.e. the model predicts if the post will get less (or as many) or more shares than the user's posts typically.

For initial experiments, a fully connected neural network with three layers (input, hidden layer and output layer) was used. Simpler models, such as logistic regression (linear classifier), were experimented, but they provided weaker results than the neural network based approach. The initial results obtained in spring 2024

by using the neural network-based approach are shown in Table 5. However, these results may still change when more data is obtained for model training and further feature engineering is done.

As a data source for PIA model development, applicability of several social media platforms were studied. Social networking service 'X' (former Twitter) was assumed to be an ideal data source for the task, as it has also been used in earlier studies found in the literature. However, by the time of decision in Spring 2023, 'X' posed a relatively high fee to use its data, making it unfeasible. Reddit was considered as a second option, but due to changes in terms and conditions at about the same time, it also was ruled out. Finally, Mastodon was taken into use. Mastodon resembles 'X', but it provides an API without a subject to charge. As a drawback, it has less users and less content in the service. Anyway, APIs in Mastodon allow to fetch information related to user profiles and posts.

A crawler was developed to collect data from Mastodon API as chunks. In the crawler, a username is given as a starting point, after which the crawler fetches the posts for the user and the profile information and stores them. Next, it checks other usernames that the user is linked with, and these usernames are added to "to-be-checked" list. Posts and profile information for these users are fetched and "to-be-checked" list is updated with new usernames. After collecting a reasonable amount of data, this procedure is repeated to collect more chunks, and as a final result, a dataset is obtained.

In addition to the Mastodon data, we have been granted access to a corpus of data from Suomi24, which is a long-standing Finnish discussion forum. As a traditional discussion board type of social media, Suomi24 resembles Reddit more than X/Twitter or Mastodon. Furthermore, we are exploring if Meta's new Threads platform might be usable for training the PIA model. The training data for the PIA model is being scrutinized for privacy. If successful, an anonymized version will be published alongside the PIA component, as its training data.

Table 5. **Confusion matrix of the initial PIA model.** summarizes initial results (spring 2024) of the PIA model on a test dataset with the neural network based model. In this case 72% of posts that were shared less or equal amount than usually in real life were predicted correctly. 28% of the posts that were shared less or equal amount than usually were incorrectly predicted to be shared more than usually. 8% of the posts which are shared more than usually were incorrectly predicted to be shared less or equal amount than usually. 92% of posts which are shared more than usually were predicted correctly.

		Predicted class	
		Less than usual (or equal)	More than usual
Actual class	Less than usual (or equal)	4810 (72%)	1886 (28%)
	More than usual	196 (8%)	2310 (92%)

*Table 5. Confusion matrix of the initial PIA model.*

After the initial results further development has been done in an iterative manner based on experimentation and findings on relevant research. More attention has been also put on the preprocessing of the dataset. Initial experiments were done by using all of the collected data in the dataset. However, the collected Mastodon dataset contains also short posts or e.g. posts with an image with only a couple of words. These were then excluded, since they are not relevant in the TITAN scenario. At the moment, the assumption, based on further experiments, is that less optimal results (lower accuracy) are obtained with a larger curated dataset than the

ones obtained with the initial model. This is why further feature engineering and development for the machine learning model is required.

Currently (August 2024) a mechanism is being developed which allows the PIA component to use multiple machine learning models. If the social media (Mastodon) profile information is available, the classifier for PIA is selected from a pool of classifiers based on median number of shares of the user (i.e. typical amount of shares). The pool consists of classifiers trained for different median numbers of shares (i.e. user groups) and they predict if the post at hand will get more shares than the median. In case the social media profile information is not provided, a model is used without the social media profile derived features and a classifier is used that predicts if a post will get more shares than the median in the whole training set (i.e. the most typical median among the training set users).

#### 4.8 Collaboration Environment

The challenge of the collaboration module is to enhance a group's ability to critically examine a potentially disinformation-laden topic by involving different perspectives and potential disagreements in a productive discussion. For this purpose, computer-assisted argument mapping was chosen as framework of reason-based deliberation by the community. As described in Deliverable Report D2.1 "TITAN Socio-technical Framework and User Needs Analysis", section 3.4.2. In "Improving critical thinking with collective argument mapping in a collaborative module several studies have found that argument mapping is an effective method for improving critical thinking. The argument mapping method is especially beneficial in working out solutions to problems and controversies. It focuses on formulating reasons and on logical relations between them instead of addressing or attacking a person who utters their opinion. Any claim that users want to investigate can be formulated as a thesis that can be supported or undercut by other premises. In this process, additional theses open to investigation are created. Additionally, user can rephrase the theses or provide sources of their origin. All those relationships are represented on the map with different colors of arrows connecting theses in boxes. This creates a transparent, visual representation of reasoning that can be further developed by the collective intelligence of the group. Thus, argument mapping facilitates an informed and respectful discussion. Emphasis on validation of opinions and on fact-checking enhances the quality and credibility of this type of discourse. Confrontation with differing perspectives develops analytical and critical skills and is useful in consensus-building.



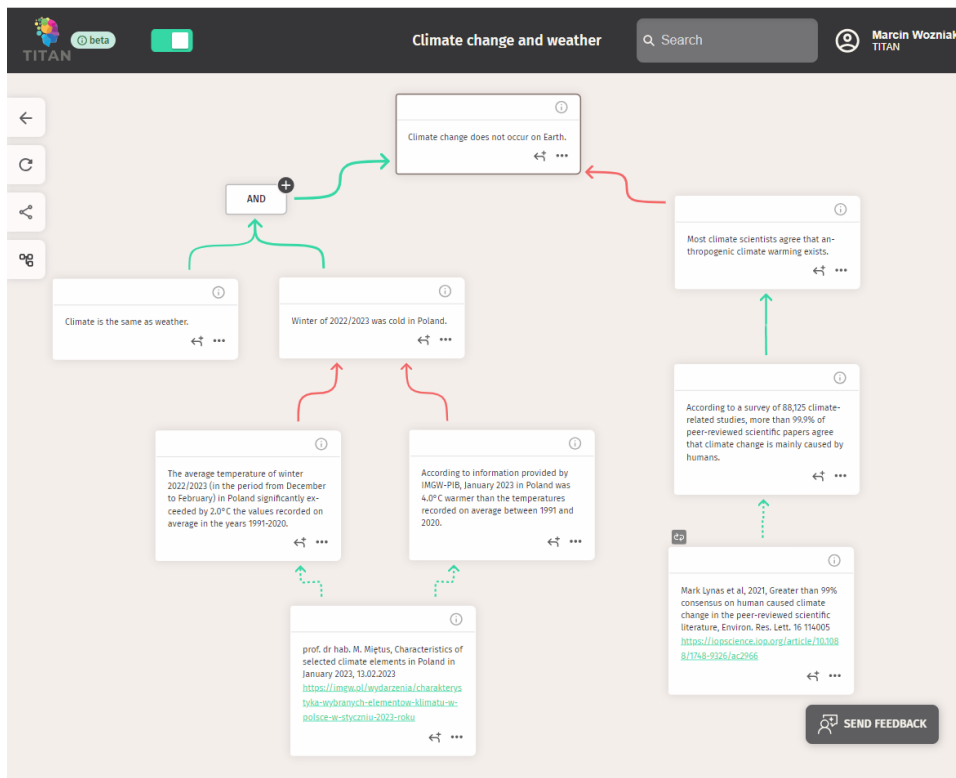


Figure 9. - Interactive argument map with claims, sources and relationships

Visual representation of argument structure using box-and-arrow diagrams showing contentions, supports, objections, conjunctions, sources etc.

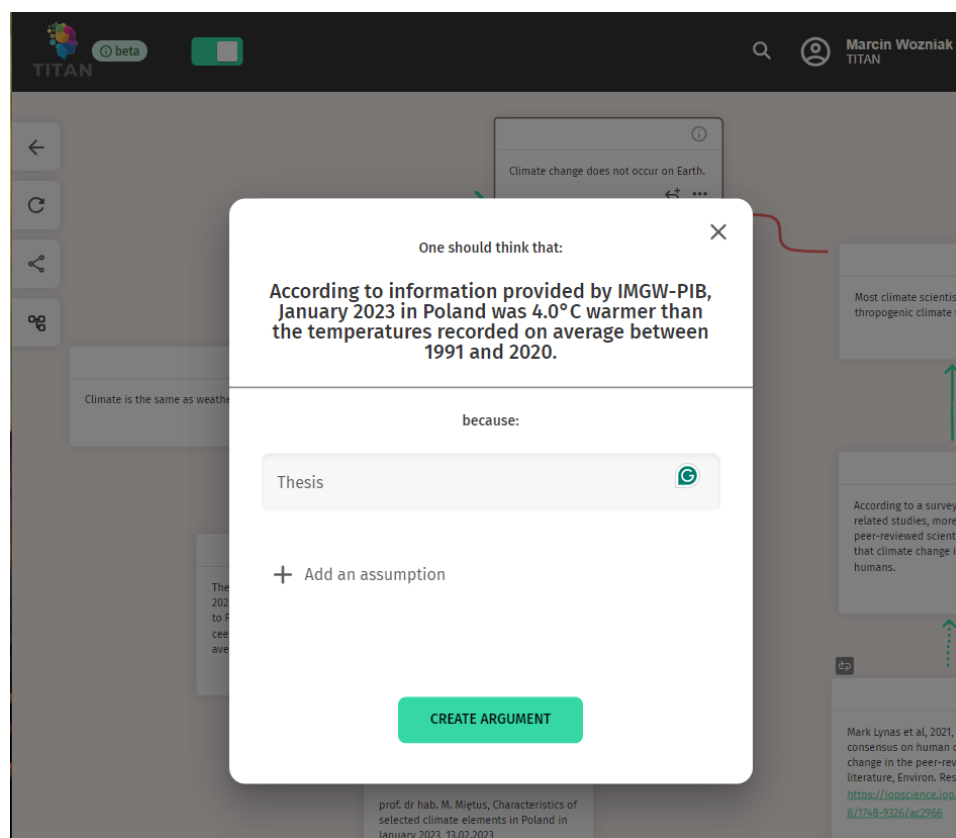
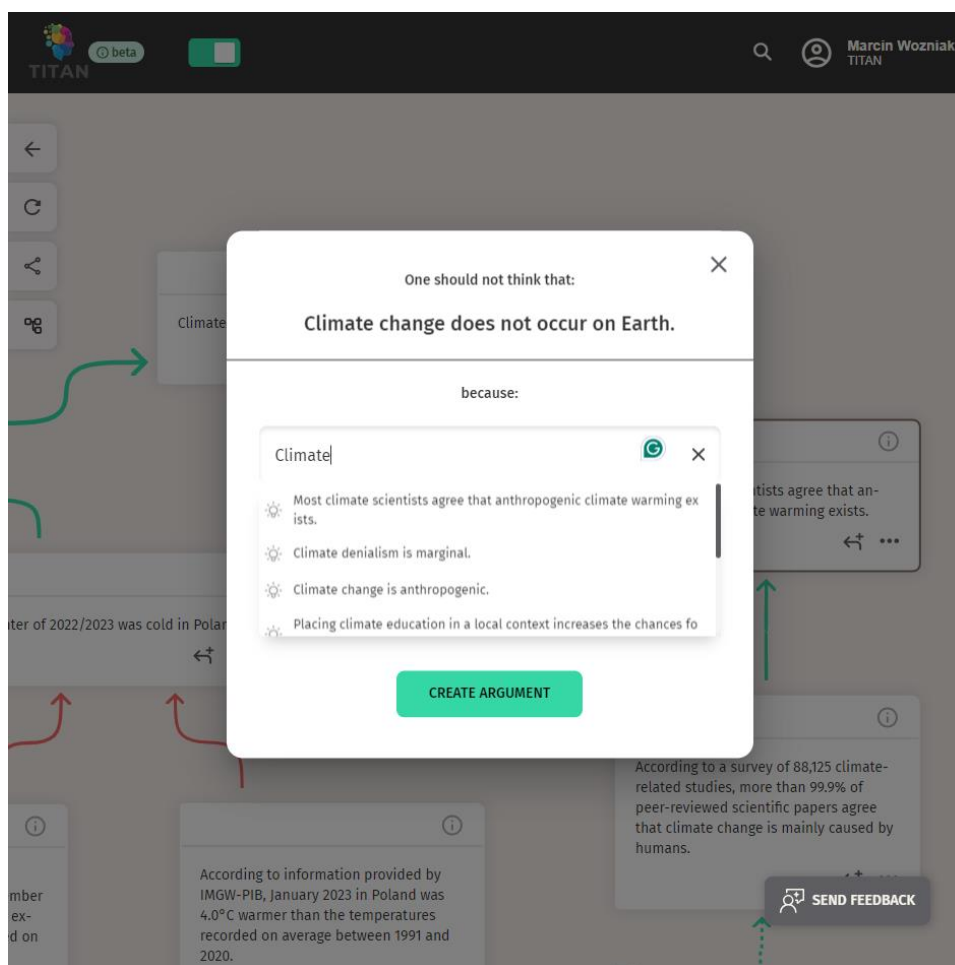


Figure 10. - Input box for constructing a supporting argument

Collaborative argumentation promotes nuanced analysis and nonpartisan examination of claims.



*Figure 11. Reusage of previous claims (theses) in input box*

To create construction of shared knowledge and true collaboration, the collaboration module supports reusage of previously stated claims and their premises. Doing so connects two graphs together. This way every claim inherits it's supports, objections, sources and related chains of reasoning.

Interactive argument maps, can share the content of argumentation, but can be divided into different discussions as well, for example to focus only on some specific claim or topic. Those topics will be displayed in TITAN app as a list of discussions.

Considering the feedback from the Living Labs and Pilot use cases reviews have significantly improved the functionality and usability of the Collaboration Environment:

- Improved Visuals: The display of arrows has been fixed to enhance the readability of the map. This improvement helps users better understand the logical connections between different arguments and claims.
- Streamlined Argumentation Options: To reduce cognitive overload, the number of argumentation options has been reduced. The basic discussion options now include support, challenge, reformulate, and source. This simplification allows participants to focus on the core aspects of argumentation without being overwhelmed by too many choices.

- Educational Materials: New tutorials and materials have been created and implemented to explain good practices in argumentation. These resources are designed to help users develop their argumentation skills and engage more effectively in discussions.
- Enhanced Discussion Context: Additional options for presenting the purpose of the discussion and additional descriptions have been added. This provides participants with a clearer understanding of the discussion's objectives and context, facilitating more focused and meaningful interactions.
- Socratic and Critical Questions: The development of Socratic and critical questions aids users in reading and argumentation. These questions encourage deeper analysis and critical thinking, helping participants to explore different perspectives and challenge assumptions.
- User Interface Improvements: Bugs in the user interface have been corrected to provide a smoother and more intuitive user experience. This ensures that participants can focus on the content of the discussion without being distracted by technical issues.
- Functional Emoticons: New functional emoticons have been created, allowing users to add simple reactions with substantive meaning, such as indicating the need to add a source or justification, expressing that a thesis changed their mind, or highlighting the importance of a point.
- Reusable Argumentation Database: Work is underway to enhance the database of prepared and verified reusable argumentation, which will be used to refute popular errors. This resource will provide participants with access to well-constructed arguments that can be used to address common misconceptions and enhance the quality of discussions.

## 5 NEW DATASETS CREATION AND CURATION

The acquisition and curation of datasets in TITAN have a crucial role in ensuring the provision of high-quality data required for fine-tuning the AI algorithms employed within the platform. This chapter delves into the methodologies and objectives behind creating and curating these datasets, with an emphasis on employing privacy-preserving co-creation methods. The benchmark datasets are designed to minimize human intervention during the co-creation process. These datasets are essential for adapting, improving, and training the TITAN AI algorithms, ensuring their robustness and reliability. Furthermore, publishing these datasets aims to encourage further research and innovation within the AI community. The objectives of the dataset creation and curation process include: compiling datasets for Socratic Dialogues to facilitate the development of AI models capable of engaging in meaningful and educational conversations (Objective 4.1); developing datasets for Micro-lessons to support the creation of concise and effective educational content (Objective 4.2); extracting datasets for identifying disinformation signals to enhance the AI's ability to detect and mitigate the spread of false information (Objective 4.3); assessing datasets for evaluating citizen critical thinking skills to contribute to the development of tools that promote analytical and reflective thinking (Objective 4.4); and evaluating datasets for assessing the propagation impact of disinformation to aid in the understanding and mitigation of its effects on society (Objective 4.5). Through these objectives, the TITAN Platform aims to advance the field of AI by providing high-quality, privacy-preserving datasets that support the development of innovative and effective AI solutions against disinformation.

### 5.1 Disinformation Signal Datasets

The 4.3 objective of our research was to gather and develop benchmark datasets for detecting different types of disinformation signals in text segments such as hate speech, offensive language, and clickbait as well as fact-checking and logical fallacies. We started by collecting all the open-source annotated datasets available online in English that were related to our research.

We initially focused on hate speech and offensive language, as well as clickbait in English texts. Datasets containing the same disinformation signal were compiled to create a comprehensive collection. The next step was to pre-process the datasets, which involved applying techniques such as data cleaning and removing duplicates. Data cleaning was necessary to remove any irrelevant or incomplete data and to ensure that the

datasets were consistent and accurate. Removing duplicates helped to eliminate any redundancy and increase the overall quality of the dataset.

Once the datasets were pre-processed, they were used to fine-tune Transformer-based Language models that could accurately detect the disinformation signals of hate speech, offensive language and clickbait, respectively. These models can automatically identify and flag any instances of disinformation in English text.

In the future, we plan to expand our research by combining and pre-processing the datasets of fact-checking and logical fallacies to develop machine-learning models that specialise in detecting and verifying facts and identifying fallacies in arguments, respectively.

## 5.2 Disinformation Signal: Hate Speech & Offensive Language Binary Class Datasets

The rise of hate speech and offensive language on social media platforms has become a major concern in the digital era. Hate speech is directed towards specific individuals or groups with the intention of being derogatory, humiliating, or insulting. Offensive language can be more general and does not necessarily target a specific group. It can be seen in offensive comments during user interactions or posted multimedia content.

Detecting hate speech and offensive language automatically is challenging because of the nuanced and context-dependent nature of these forms of communication. To address this issue, separate text classification models for hate speech detection and offensive language detection have been developed. These models leverage machine learning techniques to learn patterns in the data and differentiate between hate speech and offensive language.

Since a text can be both hateful and offensive or only one of them, developing separate models allows for more precise detection and categorization. This is crucial for effective moderation and intervention strategies. However, to train these models, a diverse range of datasets is required. Each dataset offers a unique perspective on the problem, capturing different aspects of hate speech and offensive language. By collecting and combining these datasets, we can create comprehensive and representative training sets that cover a wide spectrum of hate speech and offensive language instances.

To this end, five and two open source annotated English datasets were collected for hate speech and offensive language detection, respectively. In the following paragraphs, we will delve into the specifics of each dataset.

For both hate speech and offensive language, the dataset created by Davidson et al. (2017) was utilized. It consists of tweets collected using hate speech keywords from a crowd-sourced hate speech lexicon. There are 24,802 tweets containing hate speech terms, annotated by CrowdFlower (CF) workers. Each tweet was annotated as “hate speech” (1,430 examples) or “offensive but not hate speech” (19,190 examples) or “neither offensive nor hate speech” (4,163 examples). The “hate speech” and “neither offensive nor hate speech” examples were leveraged for hate speech detection, while the “offensive but not hate speech” and “neither offensive nor hate speech” examples for offensive language detection. There were no duplicate examples.

For the hate speech detection task, the dataset from the paper “Anatomy of Online Hate: Developing a Taxonomy and Machine Learning Models for Identifying and Classifying Hate in Online News Media” by Salminen et al. (2018) was also used. It consists of 3,215 binary texts, 2,364 “Hateful” and 858 “Neutral” examples. 8 duplicates were removed, and the labels were converted into 1 to indicate the “Hate” examples and 0 to indicate the “NOT” hate examples resulting in a dataset comprising 3,207 texts, 2,358 labelled as “1” and 849 labelled as “0”.

Moreover, we used a dataset consisting of Gab and Reddit posts (Qian et al., 2019). The text column of the dataset contained a mix of both hate speech and non-hate speech sentences, and in another column, it was indicated which sentences contained hate speech. The hate speech and non-hate speech sentences were extracted separately to distinguish between the two. Finally, the hate speech and non-hate speech sentences

were combined into a new dataset containing 23,270 texts. This dataset consisted of 14,456 hate speech examples (labelled as “1”) and 8,814 non-hate speech examples (labelled as “0”). The dataset contained 942 duplicates in total, which were removed.

The “ETHOS” dataset, created by Mollas et al. in 2020, was used for the binary hate speech detection task. The dataset was compiled from comments on YouTube and toxic subreddits, which were validated using the Figure-eight crowdsourcing platform. We utilized the binary version of the dataset, which consists of 998 comments and only two labels - “0” and “1” - indicating the presence or absence of hate speech content. Out of the 517 examples we used, 163 were classified as hate speech while 354 were not. No duplicate examples were found.

Additionally, the dataset from the paper “Hate Towards the Political Opponent” by Grimminger et al., (2021) containing 2,400 texts, 2,107 “Non-Hateful” and 293 “Hateful” examples, was leveraged. There were no duplicates. The labels were only converted into “0” to indicate not hate speech and into “1” to indicate hate speech.

The second dataset used for the offensive language detection task was developed by Zampieri et al. (2019) for Task 6 in SemEval-2019 on “Identifying and Categorizing Offensive Language in Social Media (OffenseEval)”. It is called “Offensive language Identification Dataset” (“OLID”). It comprises English tweets, retrieved with the Twitter API searching for offensive keywords. A hierarchical three-level annotation was applied to the tweets. More particularly, the first level of annotation was “Offensive” or “Not Offensive”, the second level was “Targeted-Insult” or “Untargeted”, and the third level was “Individual” or “Group” or “Other”. If a tweet was annotated as offensive, it could have a target or not. In case a tweet was annotated as offensive to a specific target, then the target could be an individual, a group, or other. For our approach, only the first level of annotation was leveraged, with 14,100 examples in total, out of which 4,640 were labelled as “OFF” and 9,460 as “NOT”. 33 duplicates were removed, resulting in a dataset comprising 13,179 tweets, 4,385 “OFF” and 8,794 “NOT”. The five hate speech datasets were compiled, resulting in 48,006 examples. Out of these examples, 23,947 were labelled as “1” to indicate hateful texts, and 24,059 as “0” for non-hateful texts. Only one duplicate text was found and removed. Several pre-processing steps were applied, such as removing the index, empty texts, special characters, and converting any HTML characters to their corresponding Unicode characters. Additionally, any extra whitespace was also removed. The XLM-RoBERTa’s tokenizer (Conneau, et al., 2020) was employed to tokenize the sentences and count the number of tokens in each example. The examples containing more than 500 tokens were removed as the Transformer-based Language Models can process sequences with up to 512 tokens. Thus, the final version of the hate speech dataset consisted of 47,303 examples, out of which 23,804 were labelled as “1” and 23,499 as “0”.

The two offensive language datasets were combined, resulting in 36,528 examples. Out of these examples, 23,573 were labelled as “1” to indicate offensive texts and 12,955 as “0” for non-offensive texts. Two duplicates were removed. The same pre-processing procedure was followed as in hate speech, hence, resulting in a dataset consisting of 36,528 examples, out of which 23,573 were labelled as “1” and 12,955 as “0”.

### 5.3 Disinformation Signal: Multi-Label Hate Speech Detection Dataset

The combined dataset is based on 2 separate datasets in the context of hate speech, namely the CONAN (COunter NArratives through Nichesourcing) and the HKUST-MLMA (Multilingual and Multi-Aspect Hate Speech Analysis) dataset.

The CONAN is a multilingual and expert-based dataset of hate speech/counter-narrative pairs for English, French and Italian. It contains various types of metadata like expert demographics, counter-narrative types and hate speech topics. The HKUST-MLMA dataset is composed of a pilot corpus of 100 tweets per language, and comparable corpora of 5,647 English tweets, 4,014 French tweets, and 3,353 Arabic tweets. It contains information about the hostility type, the hostility directness (direct vs indirect), the target attribute (i.e. sexual orientation, disability), the target group (i.e. gay, immigrants, woman) and the annotator's sentiment (i.e.

shock\_disgust). It should be noted that only the English subset from both datasets has been used in the analysis.

To build the dataset, we use the publicly available datasets annotated with labels of Hate Speech and consisting of texts predominantly from Twitter. We subsequently merge partial results from all the different sources via a conversion and curation procedure that applied data cleaning, label mapping and relevant information extraction. After this process, we arrive at the 5 labels of interest and a dataset of 15680 instances.

#### 5.4 Disinformation Signal: Clickbait Binary Class Dataset

In the era of technology and the internet, the issue of "clickbait" has become a major concern in the world of online content. Clickbait refers to web content that uses sensationalist, catchy or misleading headlines to attract users to click on a hyperlink and read an article. This tactic is frequently used to increase web page views and, as a result, generate more ad revenue.

Clickbait articles are often identified by their sensational language. The headlines or captions associated with clickbait content are usually exaggerated and employ emotionally charged words, making bold claims or promising shocking information. This sensational language is used to exploit human curiosity and the desire for novelty, creating a sense of urgency or intrigue that compels users to click on the content.

Nevertheless, using clickbait and sensational language can create several challenges. Firstly, it may result in poor user experience because the actual content may not meet the expectations created by the headline. Secondly, clickbait can lower the quality of an article and the content on a platform. Lastly, clickbait headlines can promote misleading or false information, leading to the spread of disinformation.

Detecting clickbait has become a significant challenge, and machine learning models seem to be a promising solution. These models can predict whether new content is clickbait or not by learning from labelled relevant data. This feature can help in filtering clickbait articles automatically, which helps users determine the nature of the content before engaging with it. By identifying and highlighting clickbait, users can make more informed decisions about the content they consume and share. This not only improves the user experience but also plays a crucial role in preventing the spread of disinformation.

Therefore, collecting clickbait data is essential for developing clickbait detection models. To achieve this, clickbait data from five publicly available English datasets were collected. The first dataset includes 32,000 examples, evenly split between clickbait and non-clickbait headlines. The examples were gathered and annotated from Chakraborty, et al. (2016). The clickbait headlines were taken from various news sites such as "BuzzFeed", "Upworthy", "ViralNova", "Thatscoop", "Scoopwhoop", "ViralStories", "WikiNews", "New York Times", "The Guardian", and "The Hindu". No duplicate text values were found in this dataset.

The second dataset consists of two files, one containing 100 clickbait and one containing 100 non-clickbait YouTube video titles. These titles were selected based on their potential to mislead viewers. The two files were combined, and 5 text duplicates were removed from this dataset. The values 0 and 1 were assigned for the "not clickbait" and "clickbait" values, respectively.

The third dataset contains 21,029 examples, 16,738 news texts and 4,291 clickbait texts. The texts were collected from the Clickbait news detection dataset from the Kaggle "InClass Prediction Competition". The dataset contains the title and text of the news as well as its annotated label, "news" or "clickbait". 138 text duplicates were removed from this dataset and the categorical "news" and "clickbait" labels were converted into 0 and 1, respectively.

The fourth dataset was taken from Kaggle's Clickbait News Detection challenge and included a train and a validation labelled set. The train set contains 24,871 examples, 14,650 texts labelled as "news", 6,473 as "other" and 3,748 as "clickbait". The validation set comprises 3,552 examples, 2,088 texts labelled as "news", 921 as "other" and 543 as "clickbait". The two sets were combined, and 138 duplicate texts were removed,



resulting in a new dataset comprising 22,562 unique examples. The categorical “news”, “other” and “clickbait” labels were converted into 0, 0 and 1, respectively.

The fifth dataset used was the “Webis Clickbait Spoiling Corpus 2022” (Hagen et al., 2022) from the Zenodo platform. It only contains clickbait posts and manually cleaned versions of their linked documents as well as extracted spoilers for each clickbait post. In addition, the spoilers are categorized into three types, namely short phrase spoilers, longer passage spoilers, and multiple non-consecutive pieces of text. The dataset includes a training and a validation set comprising 3,200 and 800 entries, respectively. It was leveraged for the Clickbait Spoiling Challenge at SemEval 2023. The training set contained 14 duplicates, while the validation set only 1. The two sets were combined and their 5 duplicate values were removed resulting in a dataset with 3,961 entries.

All these datasets were combined, and all 20,844 duplicate texts were removed. The finalized version of the clickbait dataset consists of 37,870 texts, where 20,020 are labelled as “1” (“CLICKBAIT”) and 17,850 texts are labelled as “0” (“NOT”). The dataset was split into 80% train, 10% test and 10% validation sets by maintaining the same label distribution. More specifically, the train set contained 16,016 “CLICKBAIT” and 14,280 “NOT” texts, while the validation and test sets included 2,002 “CLICKBAIT” and 1,785 “NOT” texts, respectively. These sets were utilized for training and testing the clickbait detection models in English, enabling them to learn the distinguishing features of clickbait and apply this knowledge to unseen data. dataset links are available here:

- <https://www.kaggle.com/datasets/amananandrai/clickbait-dataset>
- <https://www.kaggle.com/datasets/thelazyaz/youtube-clickbait-classification?resource=download>
- <https://www.kaggle.com/datasets/vikassingh1996/news-clickbait-dataset?select=train2.csv>
- <https://www.kaggle.com/competitions/clickbait-news-detection/data?select=train.csv>
- <https://www.kaggle.com/competitions/clickbait-news-detection/data?select=valid.csv>

## 5.5 Augmented Logical Fallacy Dataset for Large Language Models

In the development of LLM, the creation of a specialized training dataset is a fundamental step, especially when the focus is on specific tasks such as the detection of logical fallacies in conversational contexts. Our approach to constructing a training dataset was driven by the necessity to address the lack of dedicated datasets in this domain, particularly those involving logical fallacies within informational content, such as news articles.

The foundation of our dataset was built upon an existing dataset known for its comprehensive coverage of fallacious sentences, available at <https://github.com/usc-isi-i2/logical-fallacy-identification>. This dataset contains 3425 entries, each meticulously structured to capture the essence of fallacious statements. It provides detailed metadata about the types and structures of the fallacies present in the sentences. The dataset was chosen based on its relevance and potential for augmentation.

While this dataset is a valuable resource for classifying sentences into various classes of logical fallacies, it originally lacked contextual scenarios that are crucial for training LLMs to recognize fallacies in more realistic and complex textual environments, such as news articles. To address this limitation, we employed GPT-4 to enrich the dataset. This involved creating article-like contexts for each fallacious sentence in the dataset. GPT-4 was tasked with generating narratives that naturally embedded these sentences, thereby transforming each entry into a part of a larger news article.

The construction process involved the following approach: processing each fallacious statement and creating a contextual backdrop resembling a journalistic article. Then, we created a contextual backdrop resembling a journalistic article for each statement. This backdrop provided a realistic setting for the statement and helped to make it more believable. Finally, we generated complete articles that seamlessly incorporated the fallacious

statements from the starting dataset. The resulting dataset contained both the original statements and their corresponding contexts.

The dataset was structured in a format that made it easy to process and analyse. The “text” column contained the entire generated context, and indicators highlighted the presence of the fallacious sentence within. The dataset also included two additional columns: “logic\_fallacy” and “fine\_class”. The “logic\_fallacy” column contained the name of the logical fallacy contained in the sentence, and the “fine\_class” column contained the specific type of logical fallacy.

## 5.6 Socratic Dialogues Dataset

Within the context of WP2, the project involved the creation and curation of a unique corpus comprising 25 handwritten dialogues between the TITAN AI Socratic chatbot and a user. After defining the Socratic dialogue process with an initial mapping between the Socratic question types and the dimensions of critical thinking, the dialogues were manually crafted by experts from the consortium with backgrounds in linguistics and psychology. They meticulously designed them based on a set of articles (3-5 articles provided per Pilot Partner) selected by WP5. The team of experts, which involved partners from NCSR-D, IPT, VRT, and AHS, was coordinated by NCSR-D. The main objectives of enlisting experts to craft chatbot-user dialogues based on the Socratic method and reasoning were to assess the comprehensibility and feasibility of creating such dialogues, as well as to compile a benchmark dataset for training algorithms and developing rule-based and AI chatbots (Objective 4.1).

To ensure that the dialogues were comprehensive and effective, the selected articles per Pilot Partner were annotated with specified colours to signify the reasons they disseminate disinformation. For instance, if several sentences or phrases in the article were highlighted with green, it meant that the article utilizes logical fallacies like red herring or implicit assumption. Text segments annotated in purple denoted the use of hate speech as a disinformation signal. In addition to the disinformation signals, the user’s critical thinking level was also considered. Three broadly defined critical thinking levels (low, medium, and high) were assumed to exist, but only the low and medium critical thinking levels were considered during the creation of the dialogues. Users with low critical thinking levels were deemed as having limited knowledge about disinformation and minimal critical thinking, whereas users with medium critical thinking levels were assumed to possess some knowledge about disinformation with the ability to assess information.

The experts were instructed to select one article from the list of available articles collected by WP5, choose one critical thinking level (low or medium), one disinformation signal for each article and write Socratic dialogues considering as many steps as possible, from the 6 steps identified, which were clarification (step 1), challenging assumptions (step 2), evidence and reasoning (step 3), alternative viewpoints (step 4), implications and consequences (step 5) as well as challenging the question (step 6). The conversation between Socratic chatbot and user in these dialogues aimed to improve the users’ critical thinking level while enhancing their media literacy.

The researchers analysed the Socratic dialogues and extracted a set of 17 flowcharts that represent the conversation flow of a Socratic dialogue. These flowcharts depict the order and set of questions that can be asked during each Socratic step, depending on the step and the disinformation signal. The flowcharts for steps 1, 4, 5, and 6 consist of more general questions written in the corpus, thus they are common for all disinformation signals. On the other hand, the questions in steps 2 and 3 are differentiated based on the disinformation signal, hence the flowcharts are different for each disinformation signal.

Eighteen additional dialogues were formulated based on new disinformative content. These dialogues were created by experts, following a new dialogue design methodology, as described in D2.2. These dialogues are manipulation tactic-specific, for example there are dialogues concerning disinformation tactics like conspiracy



theories, trolling, discrediting, etc. These dialogues were designed with a different approach and structure compared to the initial 25 Socratic dialogues, following extensive research by experts in the Socratic method and relevant theories from Social Sciences and Humanities. Consequently, distinct flowcharts were developed for these dialogues. The new articles and flowcharts were then integrated with the existing 25 handwritten dialogue articles and utilized by users during the Pilots and Living Labs. This provided users with the opportunity to engage in two distinct styles of dialogue with the chatbot, compare them, and offer feedback for each style and methodology for creating the dialogues.

These 43 dialogues, utilized in the Pilots and Living Labs, constitute the initial version of the benchmark dataset for the Socratic dialogues. This dataset served as the training data for the first release of the TITAN chatbot, a conventional rule-based Socratic dialogue system. Subsequently, the flowcharts were instrumental in formulating the responses, intents, and stories of this rule-based Socratic dialogue system, with several examples provided in ANNEX 1. Furthermore, the additional Socratic dialogues gathered through the real-time interactions during the Living Labs and Pilots have been amalgamated with the manually crafted dialogues to develop the subsequent iteration of the Socratic dialogue benchmark dataset. This subsequent iteration is anticipated to underpin the development of the second release of the TITAN chatbot as an advanced AI chatbot, diverging from its current rule-based implementation. Presently, the first release of the TITAN chatbot is furnished with the initial benchmark Socratic dialogue dataset.

## 5.7 Argumentation Mining Dataset

Argumentation mining is a new field in NLP that aims to automatically extract structured arguments from unstructured text. Its goal is to identify argumentative structures, such as claims and premises, and understand the relationships between them. This process is important for various applications, such as opinion mining and decision-making.

Extracting arguments from natural language texts is challenging because arguments can be expressed in many ways, with different linguistic and rhetorical devices used to present and support claims. Moreover, the context in which an argument is presented can significantly affect its interpretation.

Despite these challenges, thanks to cutting-edge developments in machine learning, specifically in the area of deep learning, argumentation mining has become a reality. A well-crafted dataset is essential for effectively training and evaluating machine learning models. A dataset specifically designed for argumentation mining was compiled using approximately 1,000 data points from daily Greek news reports, with a particular focus on opinion articles due to their richness in arguments.

The annotation process adhered to straightforward principles and parameters. Each text was annotated for major claims, claims, and premises. The Ellogon annotation tool was utilized for this annotation process. The initial objective was to build a complex network of relationships between main arguments, secondary arguments, and premises, in addition to exploring the mode of persuasion (e.g., appeal to emotion, authority, etc.). However, the latter part did not prove to be very useful as it was often left empty due to the difficulty in identification.

The major claim, often associated with the main topic or title of the article, was usually unique. The major claims, as well as the individual claims, could be supported by other claims and/or premises. Conversely, it was assumed that premises are not supported by anything. A claim or premise could support more than one major or non-major claim(s). For instance, a premise could support the major claim and 2-3 other sub-claims respectively and a claim.

The project focused on two main aspects: the relationships between major claims, claims, and premises as well as the relationships among them. The relationships were categorized as follows:

- Support: A premise or claim supports another claim or major claim.
- Attack: A premise or claim contradicts another claim or major claim.
- Support-for: The author supports the major claim, and the sub-argument supports his major claim.
- Support-against: The author supports the major claim but this subclaim fights against his major claim.
- Attack-for: The author does not support the major claim and this sub-claim supports his major claim.
- Attack-against: The author does not support the major claim, but this sub-argument does not support his major claim.

Then, the well-known pre-trained Transformer model from Hugging Face, BERT (Devlin et al.,2019), was leveraged and fine-tuned on this annotated dataset for argumentation mining. This approach allowed for a comprehensive understanding of the argumentation structure within the Greek news reports.

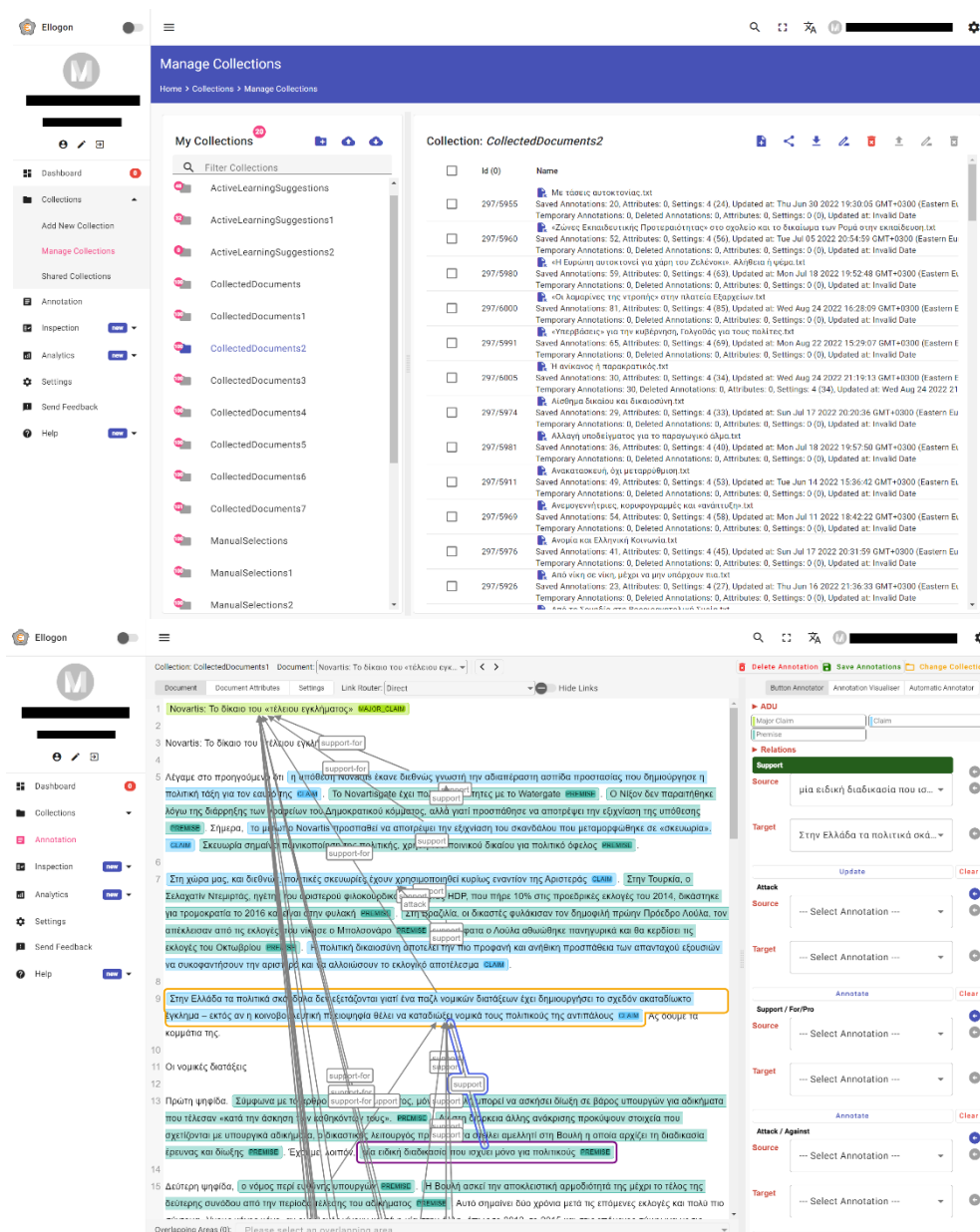


Figure 12. The annotation process in the Ellogon annotation platform.

## 5.8 Propagation Impact Assessment Dataset

As mentioned in 4.7.1, several social media platforms were considered as a data source for PIA model development, social networking service 'X' (former Twitter) being ruled out because of its pricing policies at the time of decision in spring 2023, and Reddit ruled out but due to changes in terms and conditions at about the same time. Finally, Mastodon was selected, as it resembles 'X' but it provides an API without a subject to charge, albeit having less users and less content.

A crawler was developed to collect data in chunks from Mastodon API. For a single chunk, the crawler picks up a random user, fetches the posts and profile information of the user and limited amount of their network, and stores them. While there is a speed limiter in Mastodon service, this was repeated over months with numerous independent crawler instances, resulting a reasonable sized dataset (over 2.8 million posts).

Mastodon data has been used for developing and training the machine learning model of the PIA component. The data consists of information such as:

- User account information, for example
  - Follower count
  - Number of followed users
  - Post count
  - Timestamp when the user account was created
- Information about posts done by the user, for example
  - Post id
  - Creation timestamp
  - In-reply-to-id (identifies if the message is a reply to another post)
  - URL of the post
  - Replies count (how many replies the post has)
  - Reblogs count (how many shares the post has)
  - Content (the content of the post)
  - List of media attachments in the post
- List of followers
- List of followed users

The data used for initial experiments contained ~36000 Mastodon posts. Several parallel crawlers were run for a period of few months and a dataset consisting of over 3 million posts was obtained. This dataset is being currently used for further development.

In addition to the Mastodon data, we have been granted access to a corpus of data from Suomi24, which is a long-standing Finnish discussion forum. As a traditional discussion board type of social media, Suomi24 resembles Reddit more than X/Twitter or Mastodon. Furthermore, we are exploring if Meta's new Threads platform might be usable for training the PIA model. The training data for the PIA model is being scrutinized for privacy. If successful, an anonymized version will be published alongside the PIA component, as its training data.

## 6 DOMAIN MODELLING

The TITAN Platform's domain modeling process initiated from the overarching concept, utilizing Domain-Driven Design methodology<sup>3 4</sup>. Each constituent and aspect of the framework was delineated using a shared vocabulary, establishing a universal language for design purposes. In accordance with system requirements, various sub-domains, referred to as Bounded Contexts, were pinpointed to construct the system's Data Model.

### 6.1 Ubiquitous Language

Henceforth, the definition of the system architecture will incorporate the following terminologies relevant to the domain of the TITAN Platform:

- **System User:** generic user registered on the TITAN Platform.
- **Authentication:** process to identify a user.
- **Authorization:** process to verify if a user can perform some operation.
- **Role:** role assigned to users, indicating the operation the users are allowed to perform.
- **End user:** default role for user enabled to access the Platform after registration, mostly related to final users (i.e. the citizens).
- **Super user (or Curator user):** role having additional permissions with respect of the end users, mostly related to datasets curation and microlessons uploading. Furthermore, this type of user will be also responsible to upload the Critical Thinking Assessment questionnaire for end users
- **Administrator:** role that enables the management and configuration of the whole system.
- **Permission:** policy that enable users with particular role to access stored data or system functionalities.
- **Critical Thinking Assessment:** questionnaire-based assessment to estimate the level of critical thinking of end users according to different *critical thinking dimension*.
- **Critical Thinking Assessment History:** history of critical thinking assessments done by the user over time.
- **Critical Thinking Dimension:** specific psychological or cultural aspect which is evaluated via the critical thinking assessment.
- **Critical Thinking Score:** level of critical thinking assigned to a user on a specific dimension.
- **User Content:** User provided textual content, such as articles.
- **Disinformation Signal:** A disinformation signal encompasses any communication or information deliberately crafted and distributed with the aim of misleading, deceiving, or manipulating the audience.
- **Socratic Chatbot:** conversational agent design to conversate with users according to a Socratic approach.
- **Socratic Conversation:** a conversation between the Socratic Chatbot and a user.
- **Socratic Conversations History:** history of conversations done by the user over time.
- **Socratic Chatbot Message:** a message generated by the Socratic Chatbot during a conversation.
- **Alternatives for a Socratic Chatbot Message:** the Socratic Chatbot can provide alternatives for a single response. End users may decide what alternative is more appropriate for continuing the conversation.
- **Socratic Chatbot Message rating:** end users are able to rate responses from the chatbot.
- **Socratic Chatbot Message editing:** curator users are able to edit responses from the chatbot.
- **User Message:** a message sent by users during a conversation.
- **Microlesson:** multimedia file proposed by curators and accessible to end users.

---

<sup>3</sup> Eric Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley, ISBN: 0-321-12521-5

<sup>4</sup> Scott Millett and Nick Tune, Patterns, Principles and Practices of Domain-Driven Design

- **Microlesson History:** history of microlessons done by the user over time.
- **Propagation Impact Assessment:** evaluation of the propagation of news in social media.
- **Propagation Impact Assessment History:** history of assessments done by the user over time.
- **Collaboration Environment:** interactive argument maps for conducting shared discussions with the purpose of evaluating variety of claims and reasoning.
- **Collaboration History:** history of sessions done by the user over time within the collaboration environment.
- **Generic AI Model:** Machine Learning model used to perform downstream tasks such as classification and detection.
- **AI Core Services:** A core service built around an AI model typically involves the development, deployment, and management of the artificial intelligence algorithm or model itself. This core service is responsible for processing data, making predictions, or providing insights based on the input it receives.
- **User services:** A user service dealing with users typically involves the interface through which users interact with the AI model or access its capabilities. This service focuses on providing a seamless and user-friendly experience while ensuring that users can effectively leverage the AI model's capabilities.
- **Public Dataset:** Collection of data publicly available on the web used to train and fine-tuning AI models.
- **TITAN Dataset:** dataset generated by end users and collected within the TITAN Platform to fine-tune the employed AI models.

## 6.2 Use Cases and User Roles

The roles that can be assigned to users are shown in Figure 13.

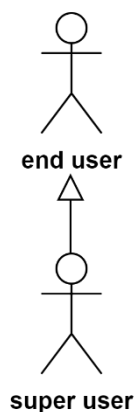


Figure 13. User Roles

Once users have verified their identity, they can commence utilizing the TITAN Platform features. These features are organized into functional domains, namely *User Management*, *Critical Thinking Assessment Management*, *Socratic Conversations Management*, *Microlesson Management*, *Propagation Impact Assessment Management*, *Collaboration Environment Management*, each catering to specific user capabilities. Upon user login, individuals can execute operations in accordance with the underlying component used and their designated role.

### 6.2.1 User Management

User management functionalities available for the different types of users' roles is reported in Table 6.

User Management	
Super User	List users
	Create users
	Assign roles to users
	Delete users
End User	View Profile
	Edit profile

*Table 6. Use cases for User Management.*

### 6.2.2 Critical Thinking Assessment Management

Super user is responsible to manage the questionnaires for the assessment of the critical thinking level of end users. More questionnaires can be uploaded, but only one can be set as *default* and proposed to end users. Table 7 shows the use cases for the Critical Thinking Assessment Management.

Critical Thinking Assessment Management	
Super User	Upload questionnaire
	Edit questionnaire
	Delete questionnaire
End User	Compile questionnaire
	View compiled questionnaire
	List done questionnaires

*Table 7. Use cases for Critical Thinking Assessment Management.*

The creation of a questionnaire involves assessing users' critical thinking skills across various dimensions. However, both the dimensions and their corresponding scores remain concealed from end users.

### 6.2.3 Socratic Conversation Management

Should there be a previous critical thinking evaluation, it will be forwarded to the Socratic Chatbot at the onset of a new conversation. Furthermore, users initiate conversations by furnishing the desired topic of discussion, specifically textual content they wish to engage the chatbot with. Disinformation signals detection is applied on the content and results are transmitted to the chatbot. Content can be provided by different means:

- An article can be selected from a list of articles proposed by curators. This use case is considered for the execution of Living Labs and Pilots, where organizers want to engage users with a controlled set of articles showing specific disinformation signals.
- An article can be extracted from a provided URL. This functionality is experimental.
- User can provide her own content.

Curator users have the privilege to modify the chatbot responses throughout a conversation, facilitating enhancements to the underlying AI model. The Socratic Chatbot has the capability to suggest microlessons to users based on the ongoing conversation. Use cases for the Socratic Conversation Management are reported in Table 8.

Socratic Conversation Management	
Super User	Edit Socratic Chatbot responses

<b>End User</b>	List conversations
	Create a conversation – select content
	Delete a conversation
	Open and continue a conversation
	Delete all conversations
	Rate a Socratic Chatbot response
	Select among alternative messages provided in a response from the Socratic Chatbot
	Open a microlesson linked in Socratic Chatbot responses

*Table 8. Socratic Conversation Management.*

#### 6.2.4 Microlesson Management

Microlessons are multimedia files organized in collections. Curators will be responsible to organize and upload microlessons, which will be then available for direct usage from users or proposed in Socratic conversations.

<b>Microlesson Management</b>	
<b>Super User</b>	Create collection of microlessons
	Delete collection of microlessons
	Edit collection metadata
	Upload a microlesson into a collection
	Delete a microlesson from a collection
	Edit microlesson metadata
<b>End User</b>	View a microlesson
	List collections
	List microlessons in a collection

*Table 9. Microlesson Management.*

#### 6.2.5 Propagation Impact Assessment Management

The purpose of propagation impact assessment management is to manage the PIA sessions performed by the end users. The end user is be able to execute the PIA functionality, view the results and manage her/his sessions. Super user is able manage sessions for all the users.

<b>Propagation Impact Assessment Management</b>	
<b>End User</b>	Perform propagation impact assessment
	View the result of propagation impact assessment
	List done_propagation impact assessment sessions for the user
	Delete a session

*Table 10. Propagation Impact Assessment Management*

#### 6.2.6 Collaboration Environment Management

The main aspect of Collaboration Environment Management is the permission level management of collective discussion on interactive argument maps. End Users must be able to make contributions to the map, but can't edit or delete contributions of others. Moderators should be able to edit all information in designated



discussion. Additionally, Administrator should be able to do anything that Moderator can, and can give moderator status to others.

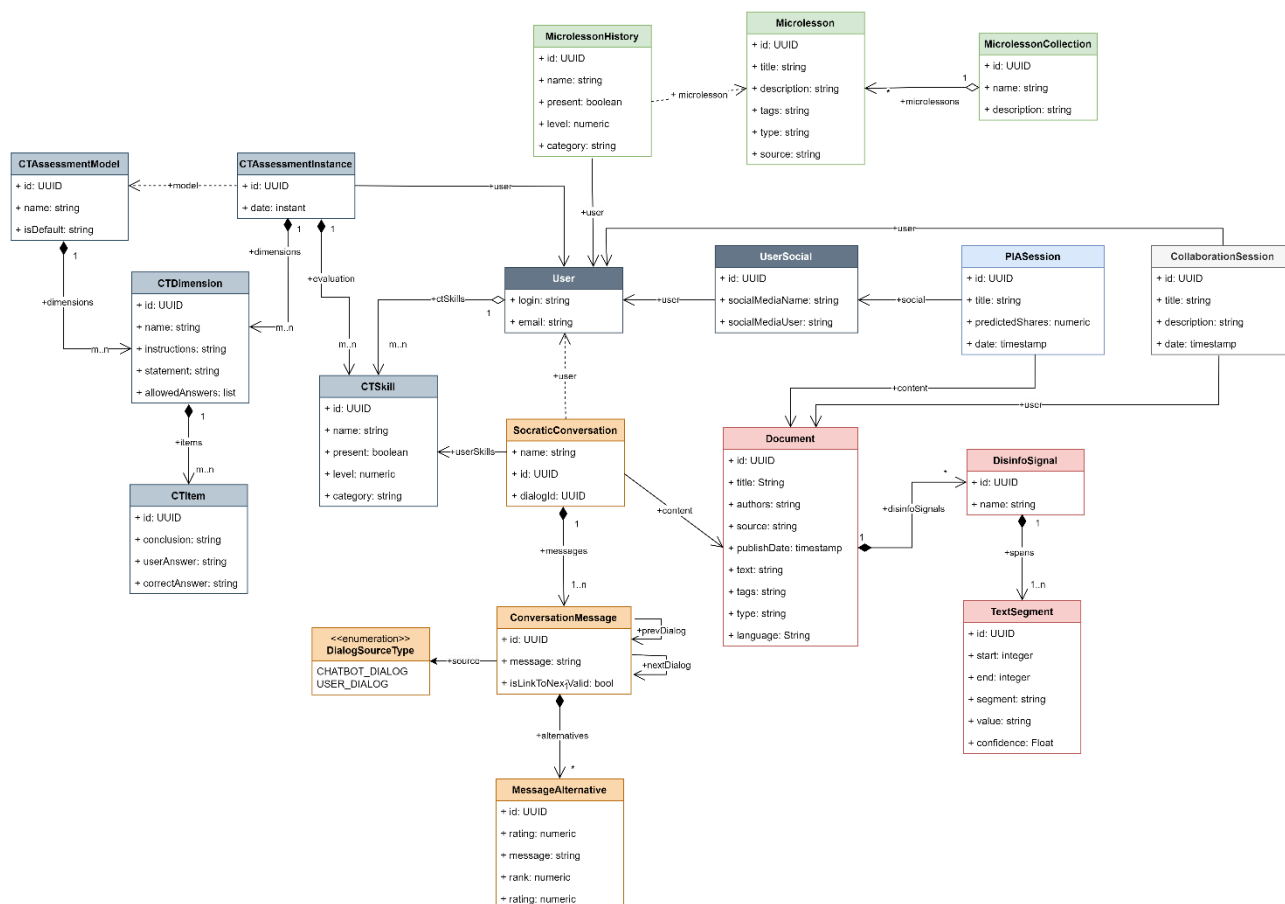
Collaboration Environment Management	
Super User	View an argument map in selected discussion
	Add claims, premises, objections, rephrases, and sources to the map
	Edit or delete any part of the map in designated discussion
	Start new discussions
	(Admin) can give Super User (Moderator or Admin) status to others.
End User	View an argument map in selected discussion
	Add claims, premises, objections, rephrases, and sources to the map
	Edit or delete their own inputs as long as other users have not used them.

*Table 11. Collaboration Environment Management.*

### 6.3 Data Model

The data and processes needed to fulfil the previously outlined use cases are structured within a domain data model, depicted by the UML diagram shown in Figure 14. Below, subsets of entities are delineated based on the functional domain to which they are linked.

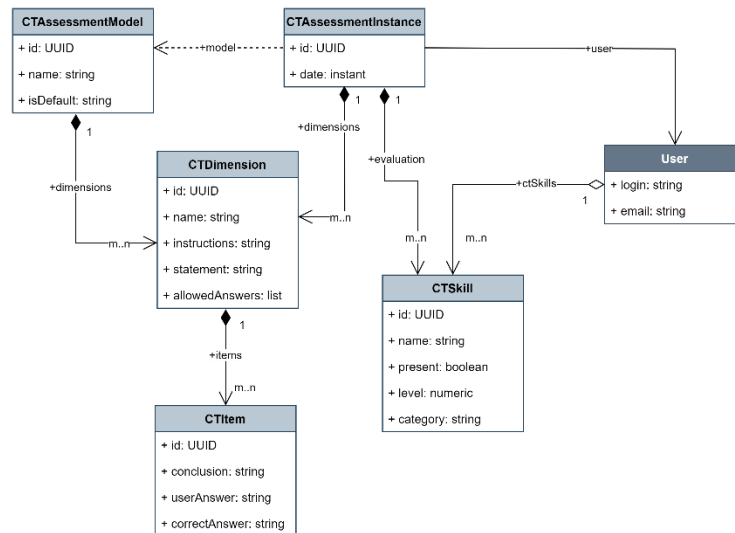
Data model will be extended to include the Collaboration Environment section during next phases of the project.



*Figure 14. TITAN Data Model version 1.0.*



Users' critical thinking assessments rely on the default assessment model specified by curators. Both the model (*CTAssessmentModel*) and the instances created for users (*CTAssessmentInstance*) consist of several critical dimensions (*CTDimension*), which, in turn, comprise critical thinking items (*CTItem*). Compiled assessments are then evaluated and linked with scored critical thinking skills (*CTSkill*). Figure 15 shows the entities related to critical thinking assessment.



**Figure 15. Entities for Critical Thinking.**

During Socratic conversations (*SocraticConversation*), the last evaluation of critical thinking (*CTSkill*), if accessible, will be utilized. These discussions consistently revolve around the content supplied by the user. The provided content (*Document*) undergoes analysis to identify disinformation signals (*DisinfoSignal*, *TextSegment*). If detected, these signals are conveyed to the chatbot to enhance the conversation. A single disinformation signal may span several lines of text. Figure 16 shows the entities related to critical thinking assessment.

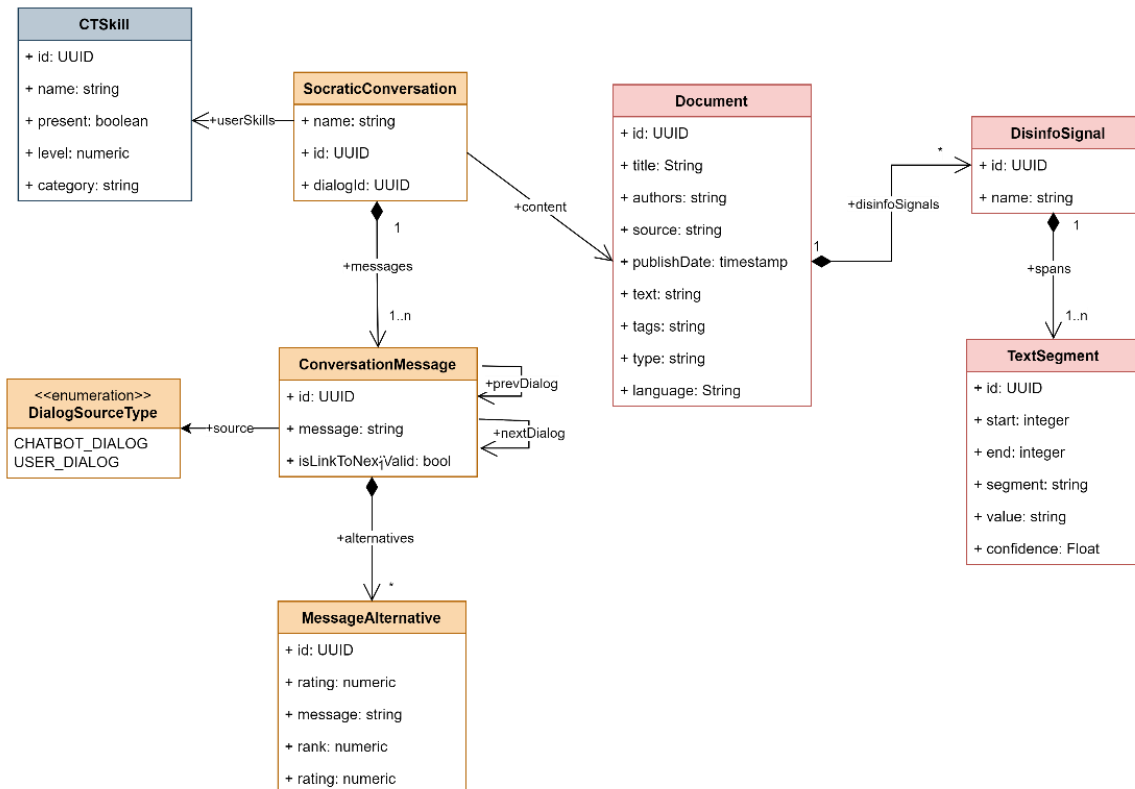


Figure 16. Entities for Socratic Conversation.

Propagation Impact Assessment and the Collaboration modules will collect the corresponding user sessions (*PIASession*, *CollaborationSession*) which involves also the selection of some content to start the session (*Document*). The PIA, also uses the social media account of the user to conduct the assessment (*UserSocial*).

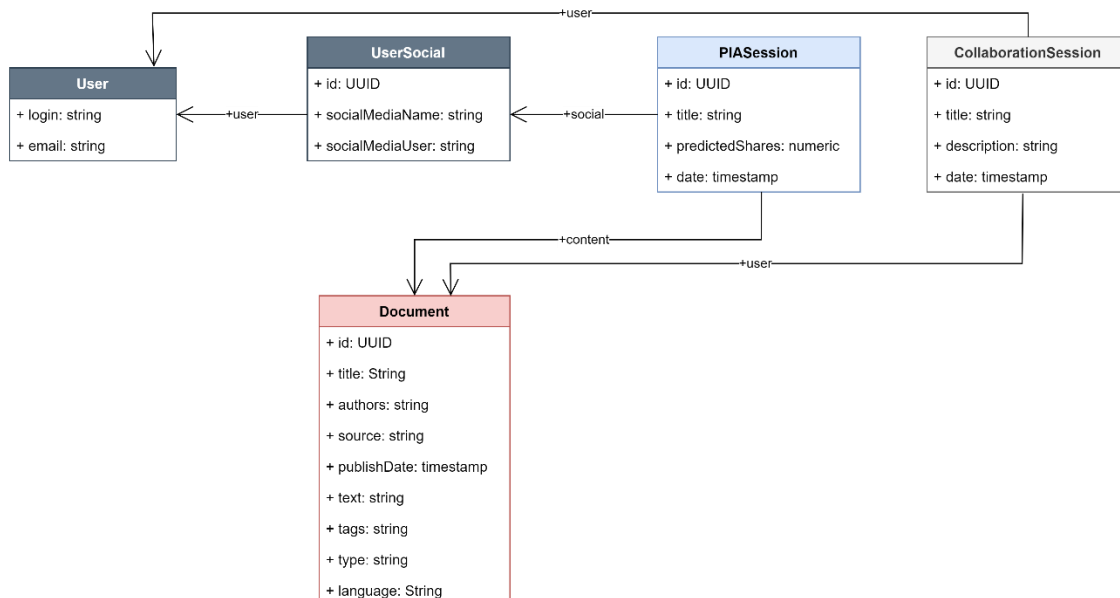
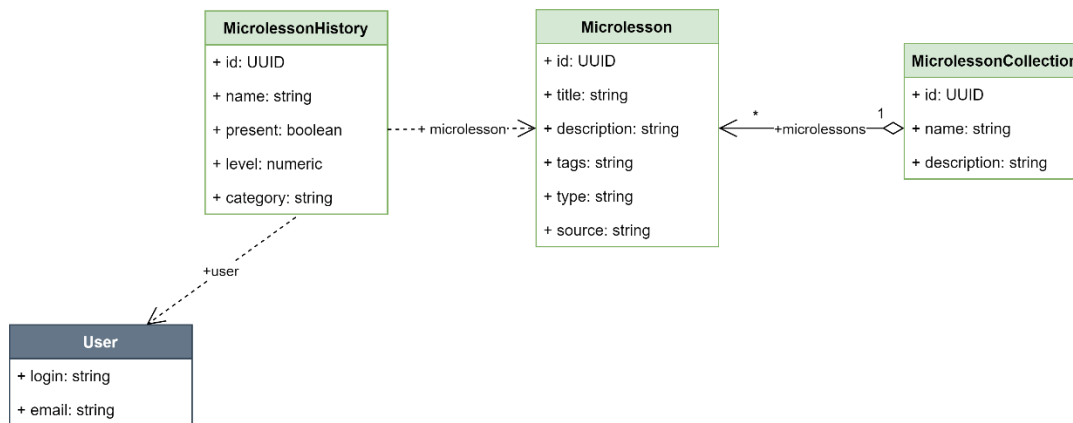


Figure 17. Entities for Propagation Impact Assessment and Collaboration.

Microlessons are organized into collections (*Microlesson*, *MicrolessonCollection*), and the system keeps the history of the microlessons consumed by users (*MicrolessonHistory*). Figure 18 shows the entities related to microlessons.



**Figure 18. Entities for Microlesson.**

Propagation Impact assessment is conducted on content provided by the user. A history of all sessions will be recorded for each user (see Figure 14).

## 6.4 Bounded Contexts

When a domain comprises several loosely connected sub-domains, the model can be partitioned into Bounded Contexts following DDD principles (BC). Each BC can be viewed as an independent set of services utilizing the same data and intercommunicating through messages or APIs.

According to the analysis of the data model the following BCs emerge:

- **User Management:** BC dedicated to managing operations such as user creation, updates, and deletions, as well as handling roles and permissions assignment, and authentication and authorization processes.
- **Critical Thinking Assessment Management:** BC enabling curators to manage assessment models, utilized by users to compile questionnaires and assess their critical thinking skills.
- **User Content Management:** BC designated to store and analyse user provided content when necessary.
- **Microlessons Management:** BC dedicated for microlessons uploading and usage.
- **Socratic Conversation Management:** BC responsible for the Socratic conversational agent, facilitating the creation, continuation, and termination of conversations.
- **Propagation Impact Assessment Management:** BC focused on interfacing users with the AI model utilized to predict the potential spread efficiency of their content when shared.
- **Collaboration Environment Management:** BC enabling creation and moderation of public discussions in different languages.

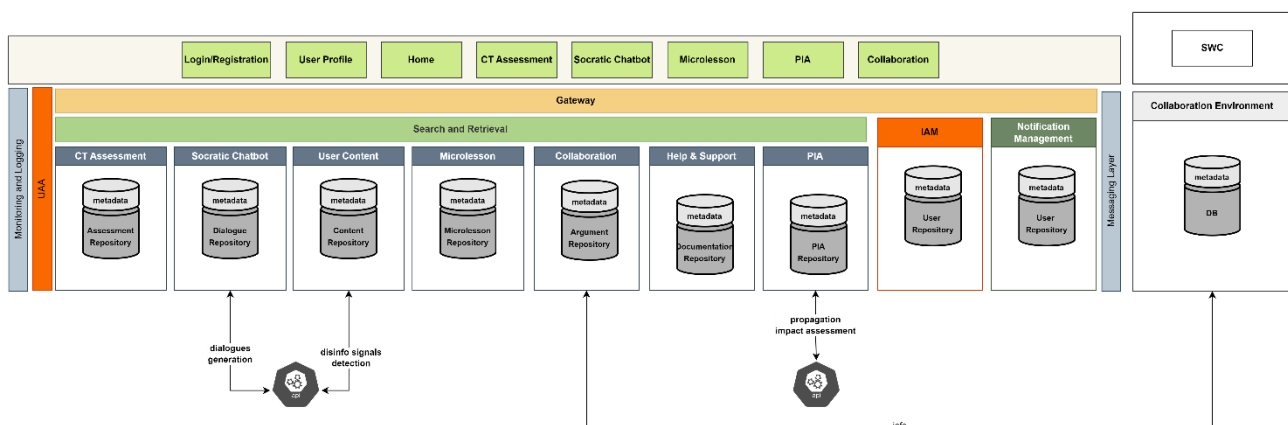
## 7 FRAMEWORK ARCHITECTURE

This section outlines the overall system architecture and the outcomes achieved through the Domain-Driven Design (DDD) approach. Additionally, it will include the mapping of architectural components with the identified Bounded Contexts (BCs).

## 7.1 Overview

To translate each functional Bounded Context (BC) into a software implementation, the Microservices Architecture has been adopted. This approach allows for the utilization of the most suitable architecture and technologies to construct each microservice according to the functionalities it is intended to provide.

The components responsible for user interaction and those responsible for providing system functionalities are categorized into the frontend and backend parts, respectively. This document primarily focuses on the backend architecture. Each frontend feature that needs to be developed aligns with the capabilities of a corresponding microservice. Given this, it would be reasonable to assume that the frontend can be structured similarly. Figure 19 illustrates the overall architecture of the TITAN Platform.



**Figure 19. TITAN Platform Architecture.**

The Gateway will route all user requests to the appropriate microservice. A dedicated Identity Access Management (IAM) service will secure user access using the OAuth2 standard. Communication between the frontend, backend, and microservices will be subject to the User Authentication and Authorization (UAA) layer. To separate the design and development of the AI model and core services from user-dedicated services, the core services built on top of the AI models can be deployed separately from the TITAN Platform services. Using agreed-upon software interfaces, TITAN services will communicate with core AI services to obtain responses. Core AI services will not store any data and will operate without persisting any data.

The Collaboration Environment will be accessed via the TITAN Platform, which will keep track of sessions created by TITAN users.

The following subsections explain the mapping of the architectural elements and the identified BCs, and the relation between frontend and backend components.

## 7.2 API Gateway

According to the microservices approach, all functionalities provided by the system are exclusively accessible through the API Gateway. This Gateway is responsible for handling requests by routing them to the appropriate microservices, which communicate with each other to produce results based on the underlying business logic.

Upon startup, microservices autonomously register with the Service Registry (SR), which maintains essential information (such as hostname, IP address, etc.) required to invoke them, even if their location changes. The SR can utilize multiple replicas of the same service for load balancing requests.

When a request is received, the Gateway retrieves information from the SR and forwards it to the appropriate microservices. If a microservice is temporarily unavailable, the client remains unaffected, and the Gateway may attempt to reconnect a predefined number of times before aborting the request to that microservice.

The use of the API Gateway conceals the details of microservices, their interaction patterns, and their locations from clients, which only need to send requests to the Gateway endpoints.

### 7.3 Microservices

This section elucidates the functionalities offered by the microservices that implement each Bounded Context (BC).

#### 7.3.1 *User Management Microservice*

The User Management microservice is responsible for all CRUD (Create, Read, Update, and Delete) processes related to managing users, assigning roles, and associating permissions for stored data and system capabilities with specific roles.

Access to the TITAN system is restricted to registered users, and only logged-in users are authorized to perform system operations. This microservice handles user authentication and authorization, maintaining an Access Control List (ACL) used to verify each external request before redirecting it to the appropriate microservice. Unauthorized operations will receive a response indicating unauthorized access.

#### 7.3.2 *User Content Microservice*

This microservice plays a pivotal role in storing and managing textual content provided by users, facilitating the utilization of various platform functionalities. Users engage with the microservice through three primary input methods:

- **Selection from Curated Articles:** Users can choose articles from a curated list provided by curators, initiating Socratic conversations based on these articles. This feature is predominantly available to Living Labs and Pilot organizers, offering controlled content showcasing specific disinformation signals.
- **URL Input:** An experimental feature allows users to input an URL, from which the content is extracted for further processing.
- **Free Text Input:** Users can also contribute free-form text, expressing their thoughts and ideas. Free text serves as the primary input for modules such as PIA and Collaboration.

Additionally, the User Content microservice is responsible for invoking the core AI service to detect disinformation signals within the content. These signals are then utilized by the Socratic Chatbot microservice to enhance its functionality.

#### 7.3.3 *Critical Thinking Assessment*

This microservice facilitates the evaluation of users' critical thinking skills through questionnaires. Users have the freedom to initiate assessments at their convenience, with a comprehensive history of all assessments retained and accessible to them for review. As per WP2 guidelines, specific scoring for each critical thinking dimension is hidden to users. Furthermore, users are restricted from modifying or deleting any assessments. The Socratic Chatbot microservice invokes the Critical Thinking Assessment microservice to retrieve the latest assessment's scoring for each dimension.

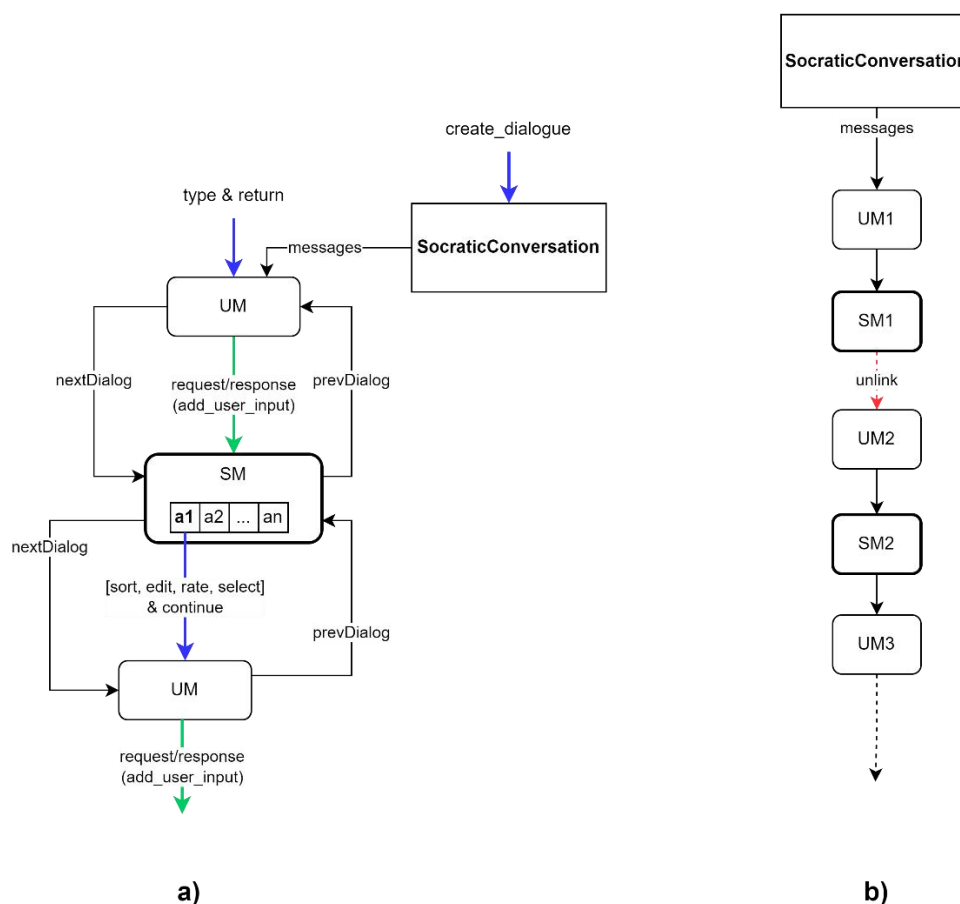
#### 7.3.4 *Socratic Chatbot Microservice*

The Socratic Chatbot microservice is responsible for facilitating Socratic conversations by providing essential functionalities to users. Interfacing with the core service of the Socratic Chatbot, this microservice is engineered to function without the necessity to store any data, as detailed in Section 4.4. Its primary features include CRUDL operations on conversations:

- **Create:** Users are able to initiate new conversations and access the entire history of past interactions.
- **Read and List:** Accessing and reviewing past conversations is enabled.
- **Update:** Conversations can seamlessly continue over different instances in time. Additionally, end users have the capability to rate messages generated by the chatbot, while curator users possess the authority to edit these messages.
- **Delete:** Users have the option to delete individual conversations or erase their entire conversation history.

In a Socratic conversation, the discussion is initiated by providing textual content to deliberate upon. The Socratic chatbot microservice is equipped to retrieve this content from the User Content microservice and analyse it for any indications of disinformation. Additionally, the chatbot retrieves the latest critical thinking evaluation of users and leverages this assessment to guide the conversation.

Conversations are managed as a linked list of messages. Upon the initiation of a new conversation, a *create\_dialogue* instruction is activated on the core service, resulting in the creation of the conversation and the delivery of a welcoming message. Subsequently, users type and send their messages (*UM - User Message*), triggering an *add\_user\_input* instruction on the core service. A Socratic message (*SM*) is then generated as a response, potentially offering various alternatives (*a1, a2, ..., an*) for the user to consider. Typically, at this stage, users have the ability to organize, evaluate, edit, and select the preferred alternative to advance the conversation. Figure 20 a) illustrate the above mechanism.



**Figure 20. General scheme of Socratic messages management.**

Furthermore, curators possess the capability to disassociate two consecutive messages within a conversation. This feature has been implemented to enable curators to rectify conversations in instances where an

erroneous response is generated by the chatbot, consequently rendering the subsequent part of the conversation invalid (see Figure 20 b).

The Socratic chatbot will also be capable to propose microlessons along the conversation.

Concludingly, this microservice will empower curators to compile datasets by downloading users' conversations. This feature will ensure the anonymization of all personal user data, prioritizing user privacy and data protection.

### 7.3.5 *Microlessons Microservice*

The Microlesson microservice is tasked with curating a comprehensive catalogue of microlessons, organized into collections. Typically, these microlessons encompass a variety of multimedia formats, including documents, images, and videos. Exclusive permissions are granted to curator users, empowering them to upload, edit, and delete microlessons as needed, ensuring quality and relevance. Conversely, end users are limited to accessing and consuming the microlessons. The microservice diligently logs the history of accessed microlessons, providing valuable insights into user engagement. Additionally, it offers robust search functionality, enabling users to efficiently navigate the catalogue using various filtering criteria.

### 7.3.6 *Propagation Impact Assessment Microservice*

Based on the data provided by the TITAN platform (free-text content, social media profile information) the Propagation Impact Assessment microservice execute the propagation impact assessment using the underlying machine learning model and return the result of estimated propagation impact via API.

### 7.3.7 *Collaboration Environment Microservice*

The Collaboration Environment microservice allow users to securely access to the Collaboration Environment, enabling them to create new public sessions or join existing ones with ease.

## 7.4 *Messaging Layer*

A pivotal component within the system, the Messaging Layer is tasked with managing various communication channels where microservices will disseminate event-related messages and receive messages from other microservices. In order to initiate message publication and consumption, each microservice must establish a connection to the messaging layer, without the need to understand the underlying technology utilized by other microservices.

The Messaging Layer comprises three primary actors:

- **Channels:** These are communication pathways utilized by producers to send messages and by consumers to receive them.
- **Producer:** A component responsible for publishing a message onto a designated channel.
- **Consumer:** A component tasked with receiving messages from subscribed channels.

By facilitating asynchronous execution of microservices, the Messaging Layer serves as a facilitator for both producers and consumers. Moreover, these roles can also be assumed by tools leveraging the messaging layer to communicate with other tools within pipelines. The Messaging Layer maintains a list of active channels and their corresponding listeners (producers and consumers), disseminating this information to microservices as needed.

Consumers have the capability to subscribe to any channel and begin receiving messages from producers publishing messages to that channel. Upon publication of a new message into subscribed channels, consumers are promptly notified, and messages destined for a consumer are queued, ensuring delivery while adhering to the order of publication.

## 7.5 Auditing and Logging

This architectural element holds a cross-sectional role within the system, impacting a majority of its components, and encompasses the following definitions:

- **Logging:** The process of recording pertinent events in a human-readable format within log files. These events typically pertain to process executions and generate messages such as "*hh:mm:ss ERROR XXX: reasons A, B, C.*" Log messages are typically categorized based on their significance, including:
  - *Errors:* Messages indicating malfunctions that affect components and/or the entire system.
  - *Warnings:* Alerts denoting errors that do not impede system operations but require attention.
  - *Information:* Messages detailing normal system functionality.

Logging primarily serves the purpose of identifying the root causes of program execution failures.

- **Auditing:** The act of recording log messages, typically at the information level, concerning specific actions performed within the system, particularly those executed by users. Auditing messages are logged in audit log files and are primarily utilized for reconstructing a user's activity, thereby identifying potential instances of system abuse, unauthorized access attempts, or improper behaviour.

## 7.6 System Monitoring

The TITAN Platform's system specification clearly identified the need for a system monitoring layer to uphold the Service Level Agreement (SLA). Given the complexity inherent in a microservices architecture, comprising numerous individual microservices, monitoring such a system presents challenges. Typically, a microservices chassis framework is employed to address this challenge, offering a suite of technologies for managing and monitoring the microservices ecosystem. This framework expedites the creation of microservices architectures by providing essential tools and functionalities, including:

- **Exception tracking:** for monitoring runtime errors.
- **Health checks:** to assess the status of microservices and components.
- **Workload monitoring:** to analyse system resource utilization in relation to connected users.
- **Default configuration:** offering preset settings and behaviours for the system.

By collecting and storing measurements from each component and applying metrics, effective system monitoring can be achieved. This approach ensures that every system component is developed with inherent measurement capabilities, facilitating comprehensive monitoring and analysis.

## 7.7 Adherence to Privacy-by-design principles

The TITAN project adheres to and implements the General Data Protection Regulation (GDPR) and Directive 2016/680/UE, the new European privacy and data protection legislative framework. These legal documents explicitly endorse the principles of Privacy by Design (PbD), initially conceptualized by the Ontario Information Commissioner Anne Cavoukian<sup>5</sup>. In the context of TITAN, these regulations apply both during the research phase and in the final use of the project outcomes.

The TITAN Consortium and project plan exemplify a commitment to data protection principles throughout project execution, encompassing the mindset of project participants and the outputs and outcomes. The specification of domain entities, system architecture modules, and communication channels, as well as information exchange, has been consistently and iteratively conducted with PbD principles.

---

<sup>5</sup> Cavoukian, Ann. "7 Foundational Principles"



Outlined below are the outcomes of this continuous and iterative process, providing insights into the application of PbD concepts within the created system architecture:

- **Proactive Approach:** The TITAN system is designed to pre-empt privacy breaches by implementing robust data protection measures. Data security mechanisms are utilized to assign permissions to users on stored data, preventing duplication, sharing, or overlapping of information between microservices. Integration of the User Management microservice, encompassing authentication and authorization mechanisms, ensures anticipatory measures against violations.
- **Default Privacy Setting:** User-centric security measures, such as authentication and authorization, are ingrained by default in the technical development of the TITAN system. Personal information is automatically set to private visibility, requiring no action from users to safeguard their privacy.
- **Privacy Integration:** Microservices handle only the data strictly necessary for desired functionality, excluding overlaps and extraneous information.
- **Unimpeded Functionality:** Privacy considerations do not impede the definition of the system architecture or the data model. Anticipated functionalities remain intact despite the implementation of privacy techniques.
- **Lifecycle Protection:** Access to data within the TITAN system is restricted to registered users with specific roles and permissions. Only users with the Super user role possess the capability to add new users, ensuring authorized access to stored data.
- **Visibility and Transparency:** A user-friendly graphical interface, complemented by tooltips explaining functionality and steps, provides users with comprehensive visibility and accessibility to TITAN system features.
- **User Privacy Respect:** TITAN offers users graphical interfaces to view both their personal data and compartmentalized information at the investigation level for Law Enforcement Agencies (LEAs).

It's essential to recognize that PbD is an ongoing process, extending beyond the system architecture design phase. It continues to inform technological implementation, services, and tools, evolving alongside the complete system.

## 8 TITAN PLATFORM GRAPHICAL USER INTERFACES

Following the iterative development and the continuous integration approaches adopted in the project to support a continuous feedback loop, The TITAN GUI is the most frequently updated component of the first major release of the TITAN Platform, adding new components as soon as they're ready for integration and incorporating the early feedback gathered during the living labs and pilots.

The incorporation of partner feedback during the mock-up stages and then during the Living Labs and pilot use cases proved instrumental in refining the design and functionality of the system. This iterative and collaborative process led to the evolution of the mock-ups' initial design, shaped by the user-journey and the user-flow elaborated based on the requirements and partner insights, presented and revised during the co-creational workshops and then implemented in the first release of TITAN, and finally refined based on feedback from Living Labs and pilot use cases.

The following sections of the document report the definition of the initial mock-ups and the current UI of the first TITAN release.

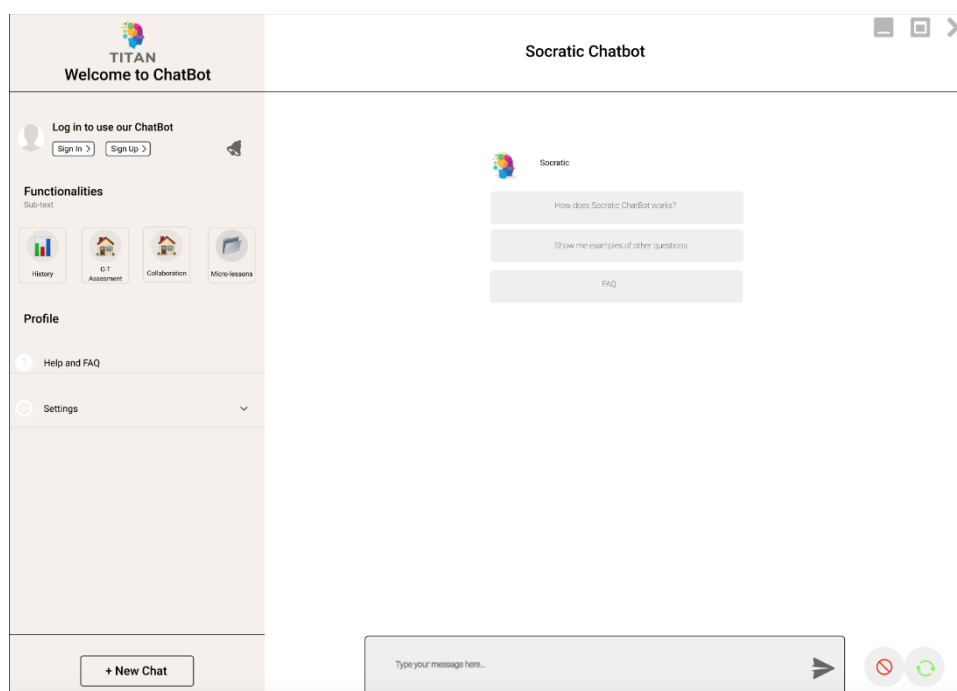
### 8.1 Initial mock-ups

The initial phase of the project involved the creation of mock-ups, a crucial step in bridging the gap between user requirements and the eventual implementation of our chatbot system. These mock-ups served as visual representations of the platform's interface, providing a tangible preview of how the system would appear and function in real-world scenarios. Grounded in user requirements, our mock-ups played a pivotal role in

exploring diverse design possibilities and refining features to ensure the system's effectiveness in meeting user expectations.

The iterative design process was integral to the mock-up creation, with multiple iterations undertaken to enhance their quality. Each iteration involved soliciting feedback from project partners, encompassing disinformation experts, digital literacy advocates, and potential end-users. This collaborative approach facilitated a comprehensive evaluation of the mock-ups, identifying areas for improvement and aligning the application design with the diverse needs, preferences, and goals of stakeholders.

In the initial deliverable (D2.1), illustrating the landing page of the TITAN platform was presented Figure 21. This visual depiction was an outcome of the first year of the project, showcasing the envisioned design. Similar mock-ups have been created for the various functionalities present on the landing page. This section aim is to present all the initial mock-ups designed in the project, offering a comprehensive overview of the visual design and functionality initially envisioned for the complete chatbot system.



**Figure 21. TITAN landing page.**

In Figure 29, a critical thinking assessment feature to evaluate users' critical thinking skills is depicted. This functionality encompasses several key components:

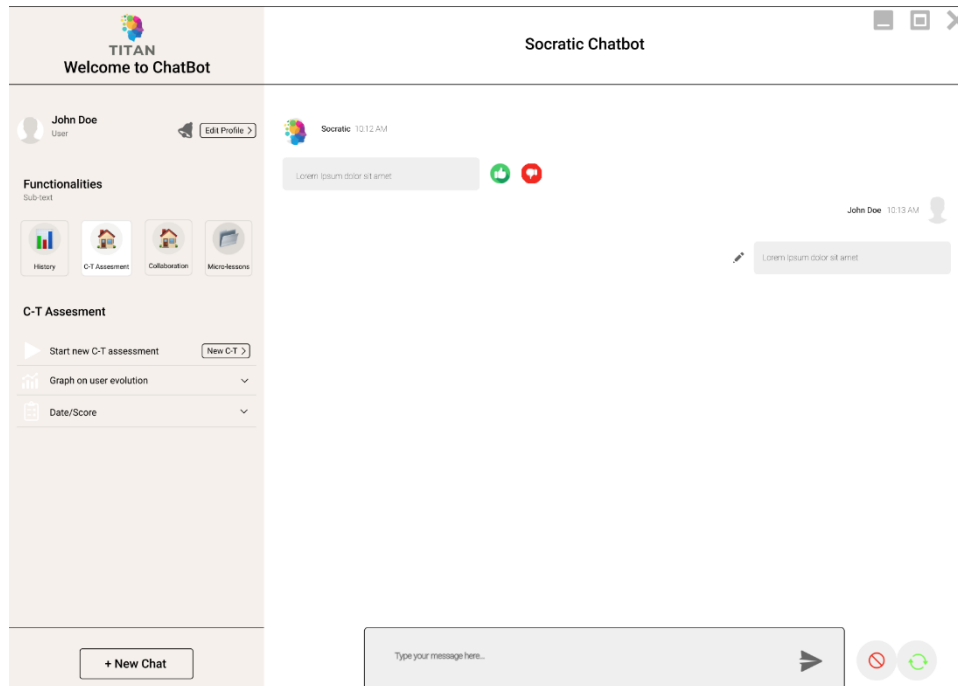
**Start new Critical Thinking Assessment:** Users can initiate the critical thinking assessment through a user-friendly interface. This streamlined process ensures accessibility and ease of use, allowing users to engage with the assessment seamlessly.

**Graph of User Evolution:** A graphical representation of users' critical thinking evolution over time provides valuable insights into their progress. This dynamic graph visually depicts how users' critical thinking skills evolve throughout their engagement with the platform. This feature not only enhances user awareness of their own development but also allows for a quick and intuitive assessment of the effectiveness of the platform in fostering critical thinking skills.

**Date/Score Tracking:** The critical thinking assessment functionality includes a tracking system that records and displays the date of each assessment session along with the corresponding scores. This feature enables users to monitor their performance over time, facilitating a more comprehensive understanding of their strengths and areas for improvement in critical thinking.

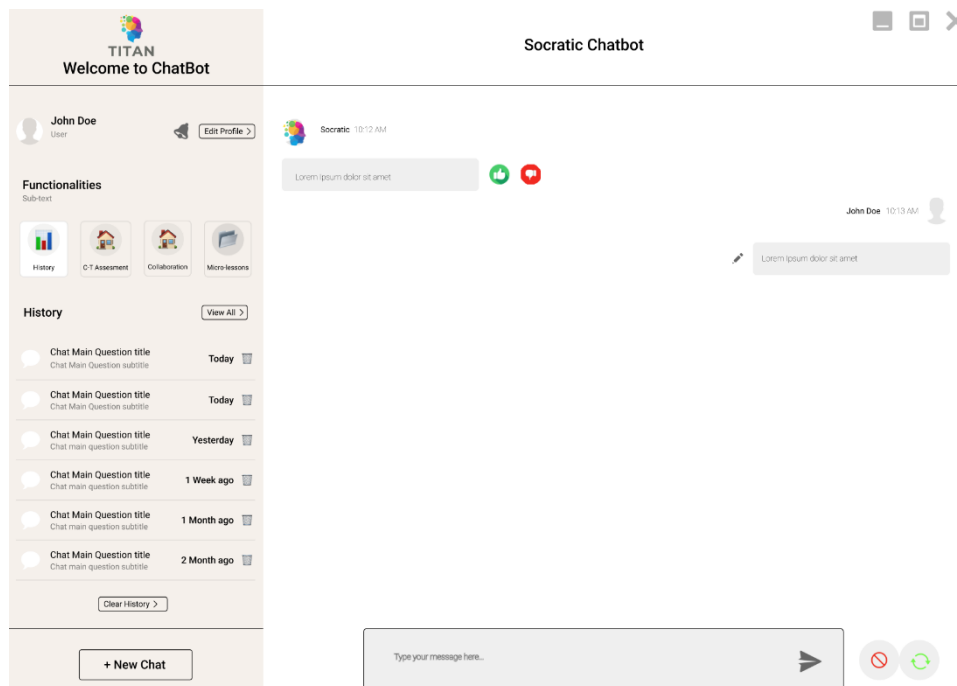
By integrating these components into the platform, the goal is to provide users with a robust tool for self-assessment and improvement in critical thinking skills. This functionality aligns with our commitment to

delivering a holistic and effective chatbot system that not only addresses disinformation concerns but also contributes to the overall cognitive development of users.



*Figure 22. Critical Thinking Assessment*

In our effort to enhance user experience and facilitate meaningful interactions, TITAN platform incorporates a comprehensive 'History' feature as depicted in Figure 23. This functionality allows users to revisit and review past interactions, fostering a seamless and personalized engagement process. Users can access a chronological record of their conversations, providing a detailed account of previous engagements with the chatbot.



*Figure 23. Socratic Conversations History.*

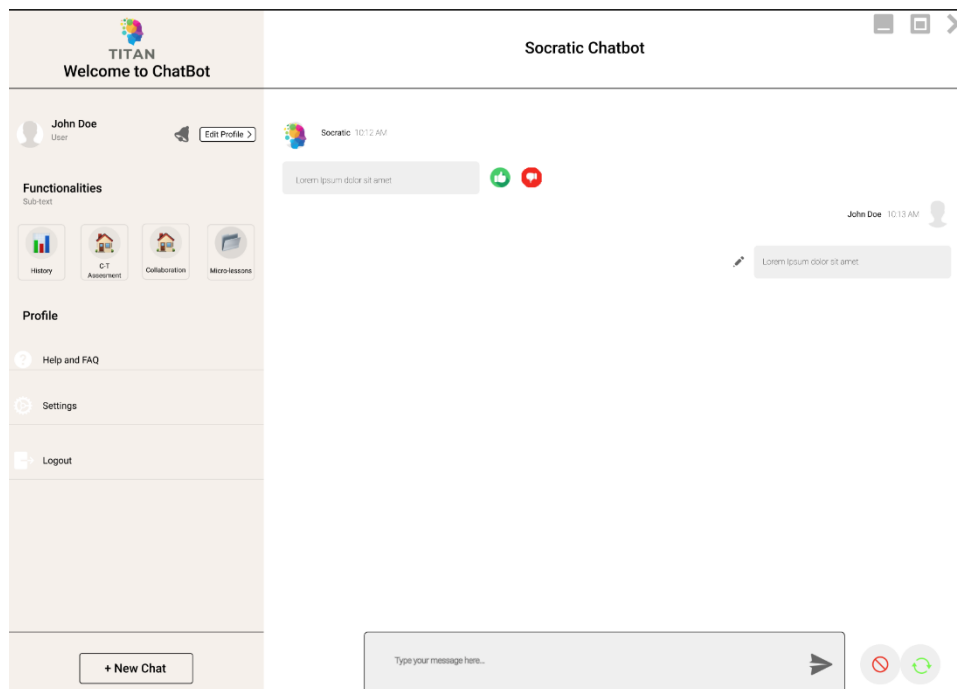
To enhance user engagement and customization, our platform features a user profile section as depicted in Figure 24 with a suite of functionalities designed for user convenience. Here's a brief overview of the key components:

**Help and FAQ:** The 'Help and FAQ' section serves as a valuable resource for users seeking assistance or clarification. It provides a curated collection of frequently asked questions along with detailed help documentation, offering users a self-service option to address common queries. This empowers users to navigate the platform independently and find quick solutions to potential concerns.

**Settings:** The 'Settings' section allows users to personalize their experience within the platform. Through this feature, users can customize various aspects (details will follow below, see Fig. 28). This user-centric approach aims to provide a flexible and accommodating environment that adapts to diverse user needs.

**Logout Option:** Ensuring user privacy and security, the 'Logout' option allows users to securely end their session and log out from the TITAN platform. This functionality is crucial in maintaining the confidentiality of user information and providing users with control over their account access.

By integrating these elements into the user profile, the aim is to create a user-friendly and adaptive platform that prioritizes user empowerment, customization, and security.



*Figure 24. User Profile Sections.*

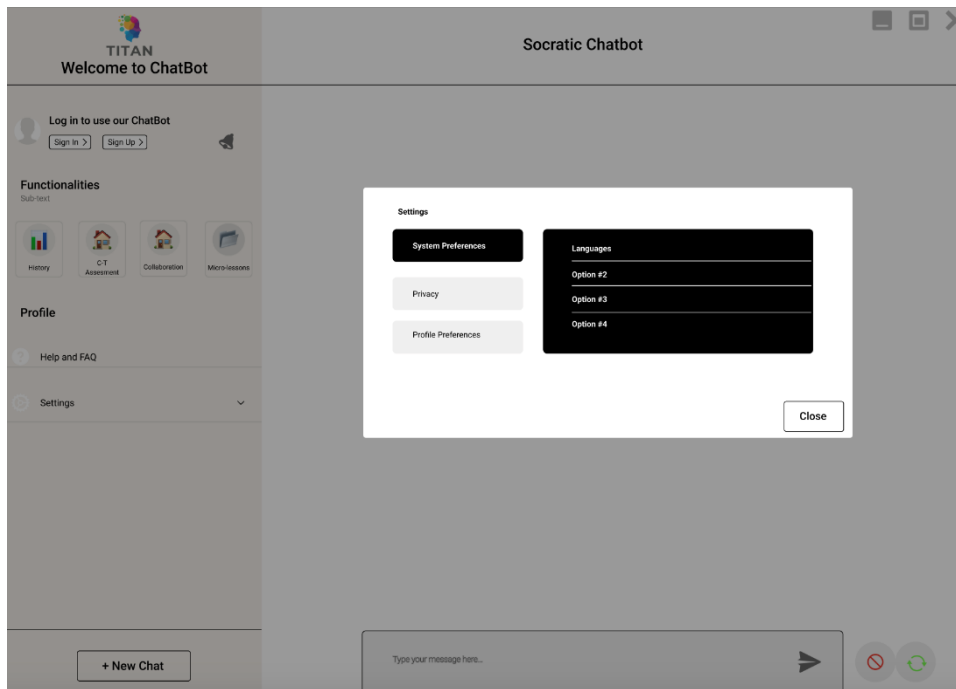
The 'Settings' option within the user profile depicted in Figure 25, is a pivotal component of our platform, offering users the ability to tailor their experience to align with their preferences and requirements. This feature encompasses three key areas of customization:

**System Preferences:** Under the 'System Preferences' section, users can personalize their language preferences and other system-related settings. This ensures a more inclusive and accessible platform by allowing users to engage in their preferred language. The flexibility in system preferences aims to create a user-centric environment that adapts to the diverse linguistic and contextual needs of our user base.

**Privacy:** The 'Privacy' section empowers users to control their privacy settings within the platform. Users can manage aspects such as data sharing, visibility of certain information, and communication preferences. This user-centric approach prioritizes privacy and aligns with contemporary standards for digital platforms, providing users with the assurance that their preferences regarding data sharing and visibility are respected.

**Profile Preferences:** This section allows users to fine-tune their profile information, ensuring that the platform accurately reflects their identity and preferences. Users can update profile details, such as display names and other relevant information. This customization fosters a sense of ownership and personalization, contributing to a more engaging and user-friendly experience.

The 'Settings' option, with its focus on system, privacy, and profile preferences, embodies our commitment to providing a platform that prioritizes user autonomy, customization, and a secure and private online experience.



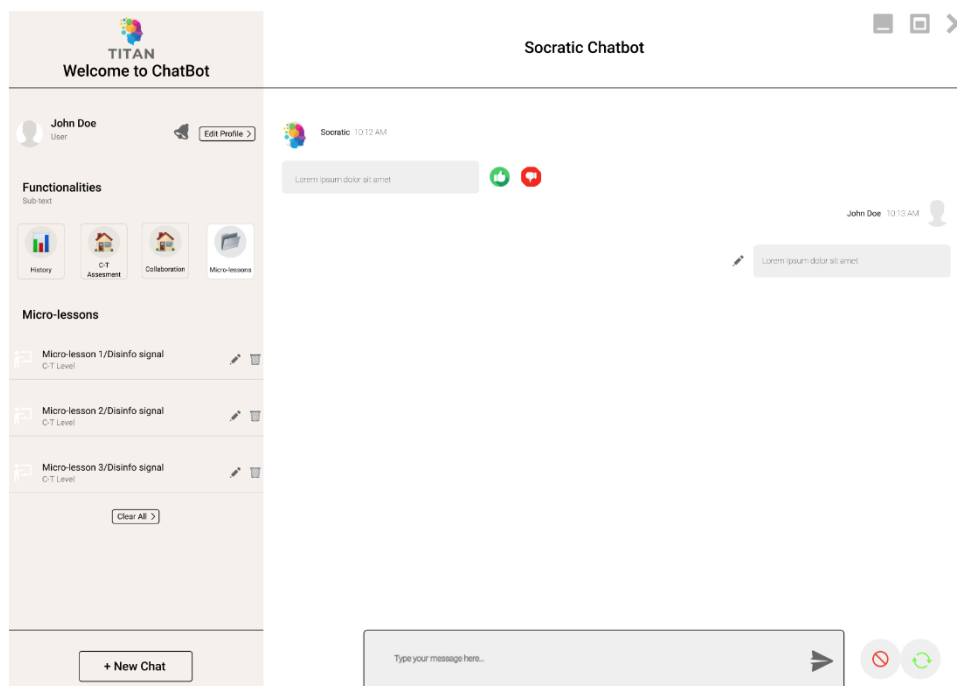
*Figure 25. User Settings.*

The 'Microlessons' tab as depicted in Figure 26, introduces a dynamic and personalized learning component, enabling users to delve into specific disinformation signals and enhance their digital literacy. Here's an overview of the key features of this tab:

**History of Microlessons Tagged with Disinformation Signals:** This feature provides users with a comprehensive record of their engagement with microlessons related to specific disinformation signals. Users can access a chronological history, showcasing the microlessons they have completed or interacted with. This serves as a valuable resource for users to track their progress, reinforce learning, and revisit relevant content, contributing to a continuous learning experience.

**User Deletion Capability:** To offer users maximum control over their learning journey, the 'Microlessons' tab includes the ability for users to delete individual microlessons. This functionality allows users to curate their learning experience, removing content that may no longer be relevant or that they have mastered. The power to delete microlessons empowers users to shape their learning path according to their evolving needs and preferences.

The 'Microlessons' tab aligns with our commitment to fostering digital literacy and providing users with practical tools to combat disinformation. By combining a history of microlessons with user-controlled deletion options, the aim is to deliver a flexible and user-centric learning experience that adapts to individual learning styles and preferences.



*Figure 26. Microlesson History.*

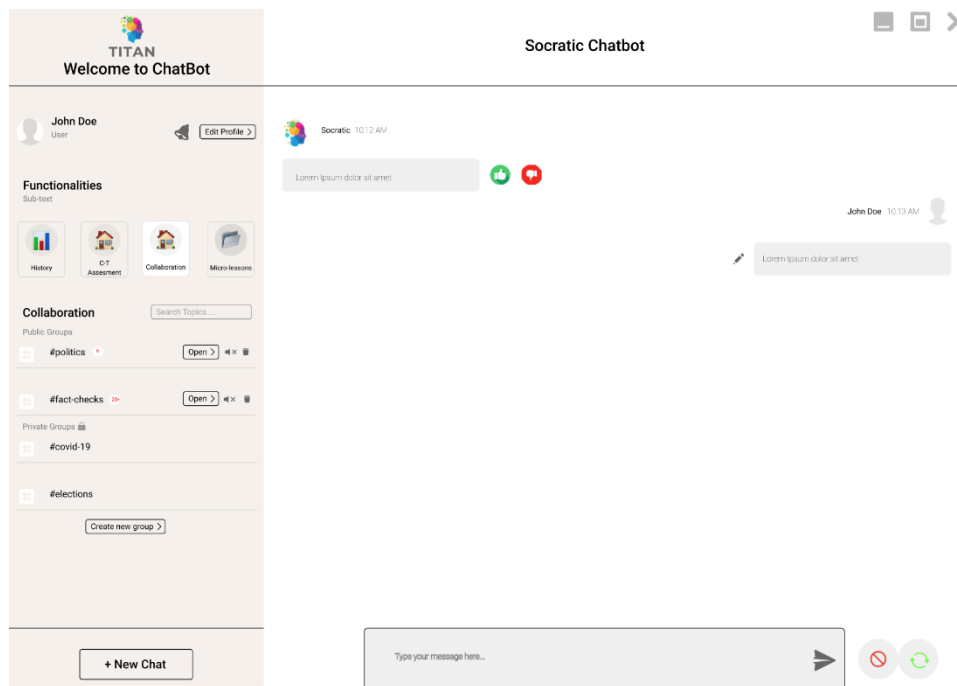
The 'Collaboration' tab depicted in Figure 27 serves as a hub for fostering meaningful interactions and knowledge-sharing among users. Here are the key features of this tab:

**Public and Private Groups:** Users can participate in both public and private groups dedicated to specific topics of interest. Public groups provide an open forum for discussions, enabling users to engage with a broader community. Private groups, on the other hand, offer a more intimate setting for focused discussions among a select group of users. This dual functionality allows users to choose the level of engagement and privacy that best suits their preferences.

**Topic Search:** The 'Collaboration' tab includes a search functionality that allows users to find and join discussions on specific topics. This feature enhances user accessibility by facilitating quick and targeted access to conversations aligned with their interests. Users can explore a diverse range of subjects, contributing to a more enriching and customized collaborative experience.

**Chat Management:** To empower users with control over their interactions, the 'Collaboration' tab incorporates options to delete or mute chats. Users can curate their chat history by deleting individual conversations, helping them manage their digital space and prioritize relevant discussions. Additionally, the ability to mute chats provides users with the flexibility to control notifications and focus on specific conversations at their own pace.

The 'Collaboration' tab reflects our commitment to building a community-driven platform, where users can engage, share insights, and learn from one another. By combining public and private groups, topic search functionality, and chat management options, the aim is to create a collaborative space that adapts to users' preferences and promotes meaningful interactions.



*Figure 27. Collaboration Session History.*

In conclusion, the initial mock-ups of our platform are a tangible manifestation of an extensive and collaborative process in gathering and translating user requirements. Crafted through iterative design methods and valuable insights from project partners and stakeholders, these mock-ups serve as a visual representation of the intersection between user expectations and the envisioned TITAN system. The inclusion of crucial features like the 'History of Conversations,' 'Profile Settings,' 'Microlessons,' and 'Collaboration' tabs underscores our dedication to providing a dynamic and responsive platform. These design elements aim to effectively address disinformation concerns while prioritizing user empowerment, customization, and the creation of a secure and collaborative learning environment. These mock-ups have proven to provide a solid groundwork for the development of the first release of the platform, with a commitment to refining features based on user feedback and the insights gained from living labs and pilot use cases.

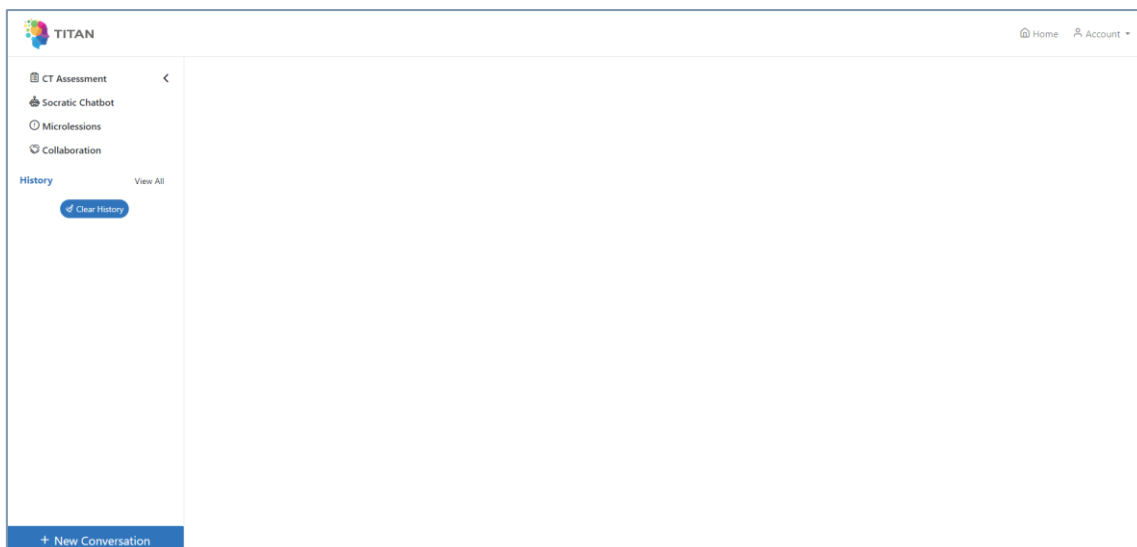
## 8.2 TITAN Platform First release UI

The current interface of the TITAN application features a user-friendly layout. Users can access the sign-in option by clicking on the Account dropdown menu located in the top right corner of the page. Upon successful login, users are redirected back to the same page. On the left side of the page, users will find the side menu, housing the core modules of the application (Figure 28):

- Socratic Chatbot
- CT Assessment
- Microlessons
- Collaboration Environment

Below these modules, there is a history item list, where users can view the most recently visited conversations, microlessons, or CT assessments. At the very bottom of the left side menu, users can find a button for creating a new conversation. This layout ensures easy navigation and access to key features of the TITAN application.





*Figure 28. TITAN Platform Home Page.*

### 8.2.1 CT Assessment

When a user clicks on the "CT Assessment" label in the left side menu, they will be directed to a dedicated page (Figure 29). The side menu includes a "CT Assessment History" section, displaying the most recently created CT assessments. To initiate a new CT assessment, the user can simply click on the button located to the right side of the label. This streamlined process ensures efficient navigation and access to CT assessment functionalities within the TITAN application.



*Figure 29. CT Assessment Section.*

Successively, user will be prompted to the page presented in Figure 30. In this section, users will encounter a list of statements, each comprising:

- Instructions
- Statement (actual content)
- Proposed conclusions

For each statement, users are required to select the appropriate conclusion based on the corresponding "Statement" section above. Once users have chosen the correct conclusion for each statement, they can

proceed to click the "Submit" button located at the bottom of the page. Upon clicking "Submit," a new CT assessment is generated.

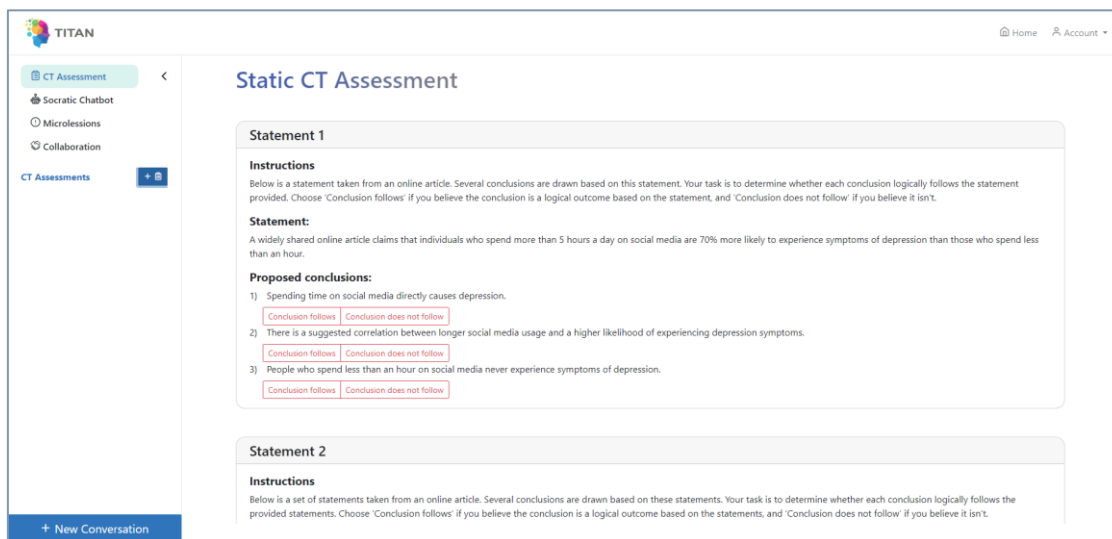


Figure 30. CT Assessment Questionnaire.

The newly created assessment will be added to the history side menu for easy access. Users can preview the assessment immediately after its creation process (Figure 31). This streamlined workflow ensures a seamless and efficient CT assessment experience within the TITAN application.

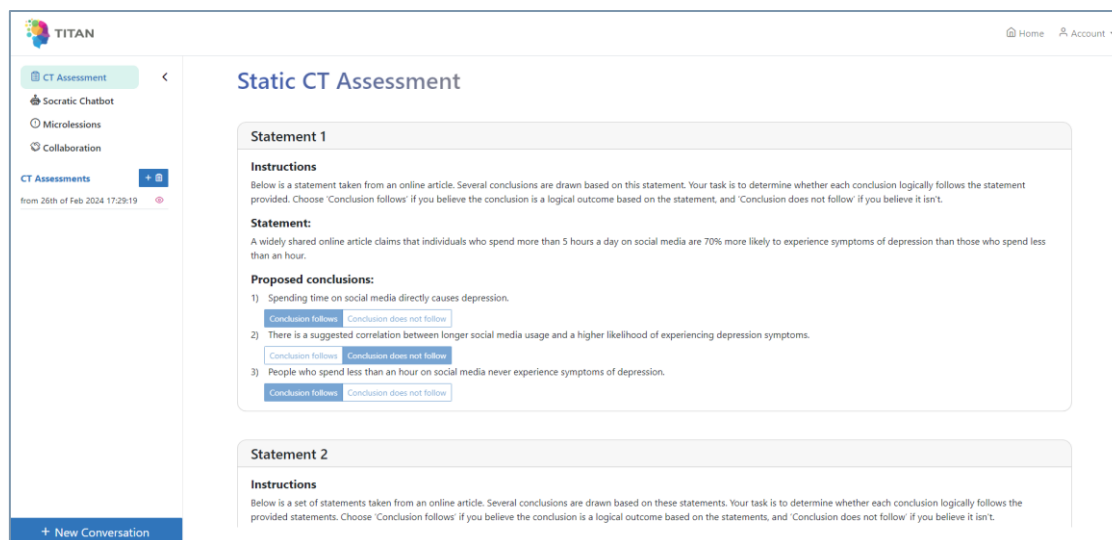


Figure 31. CT Questionnaire View from History.

### 8.2.2 Microlessons

When a user clicks on "Microlessons" from the left side menu, they will be directed to a page resembling the screenshot in Figure 32. On the left side, users can view a list of microlessons they have previously previewed. All microlessons are organized into microlesson collections, allowing users to preview the table of all microlesson collections. By clicking on a collection's name in the table, users can preview its content. Above the table, there are buttons for creating a new microlesson collection and for refreshing the page. Adjacent to these buttons, there is another button to search for a single collection using different parameters. This layout

provides users with convenient access to microlessons and their collections, along with efficient search functionality within the TITAN application.

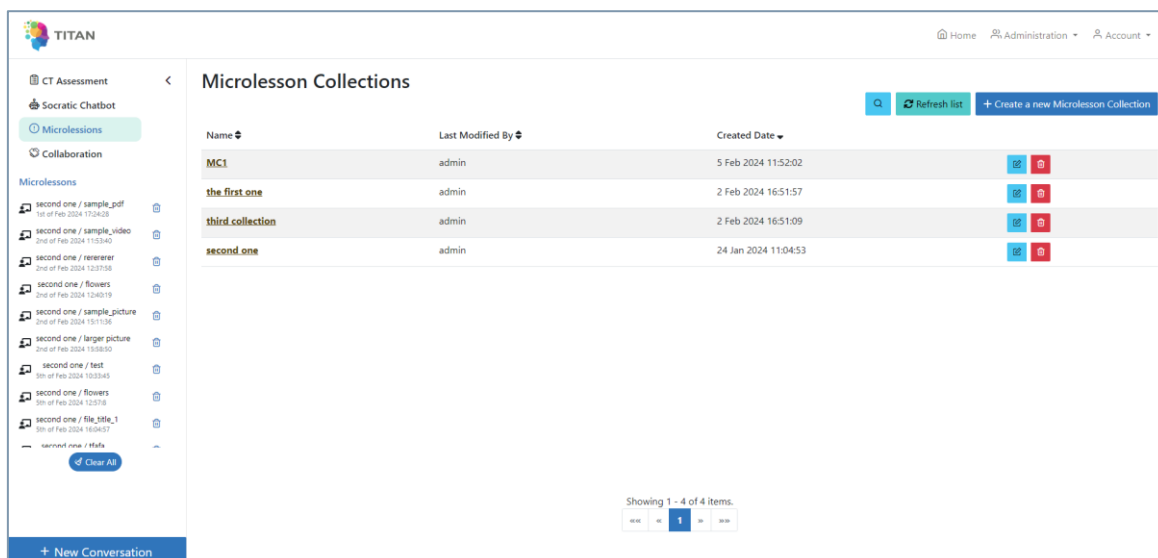


Figure 32. Microlesson Collections

Upon clicking on microlesson collection name, the user is prompted to the page shown in Figure 33:

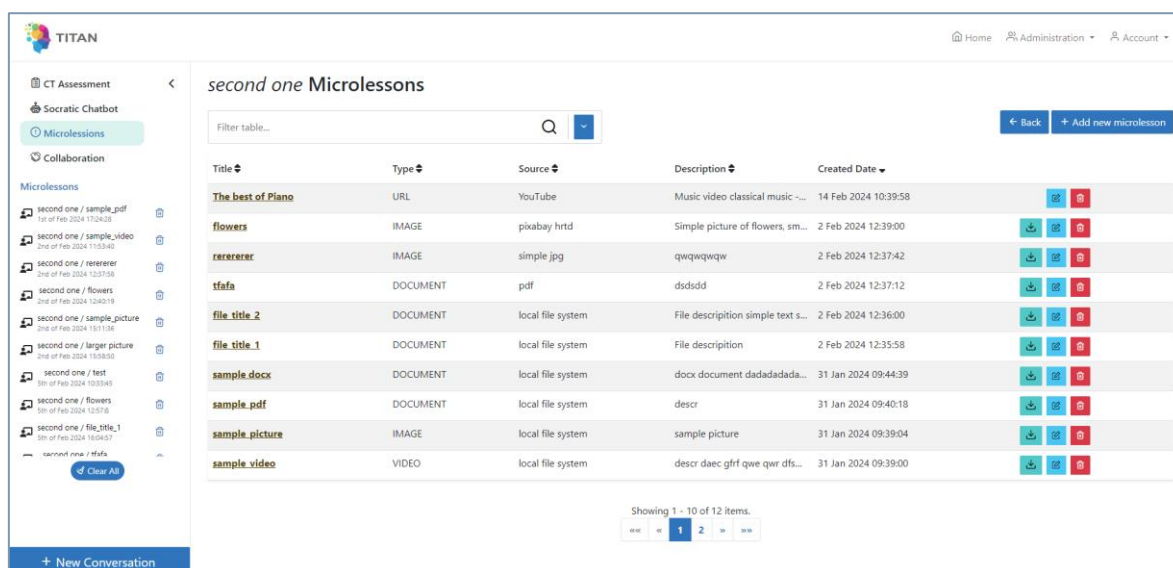


Figure 33. Microlessons in a Collection.

On this page, users can view a table displaying all microlessons that belong to the specific microlesson collection. The heading of the page indicates the name of the microlesson collection currently being previewed, providing users with clear context.

On the far-right side of the table, users with the roles ADMIN or CURATOR can access buttons for:

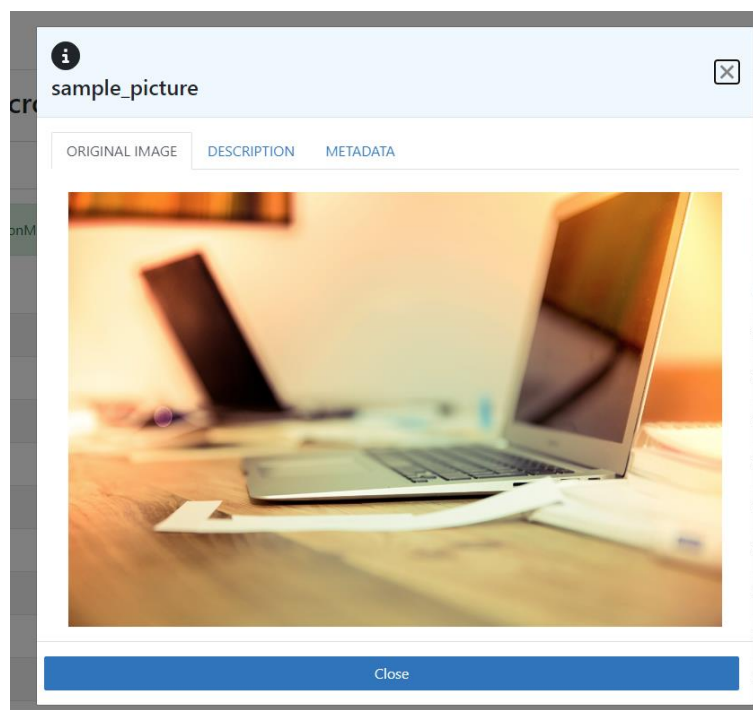
- Download
- Edit
- Delete

The table is paginated, with a paginator component located at the bottom for easy navigation. Below the page heading, there is a search bar where users can search for a specific microlesson within the currently opened microlesson collection by entering various search parameters.

At the far-right side of the page, there are two buttons:

- Add new microlesson: This operation is restricted to users with roles ADMIN and CURATOR, allowing them to add a new microlesson to the currently opened microlesson collection.
- Back: When users click on this button, they are directed back to the page displaying all microlesson collections.

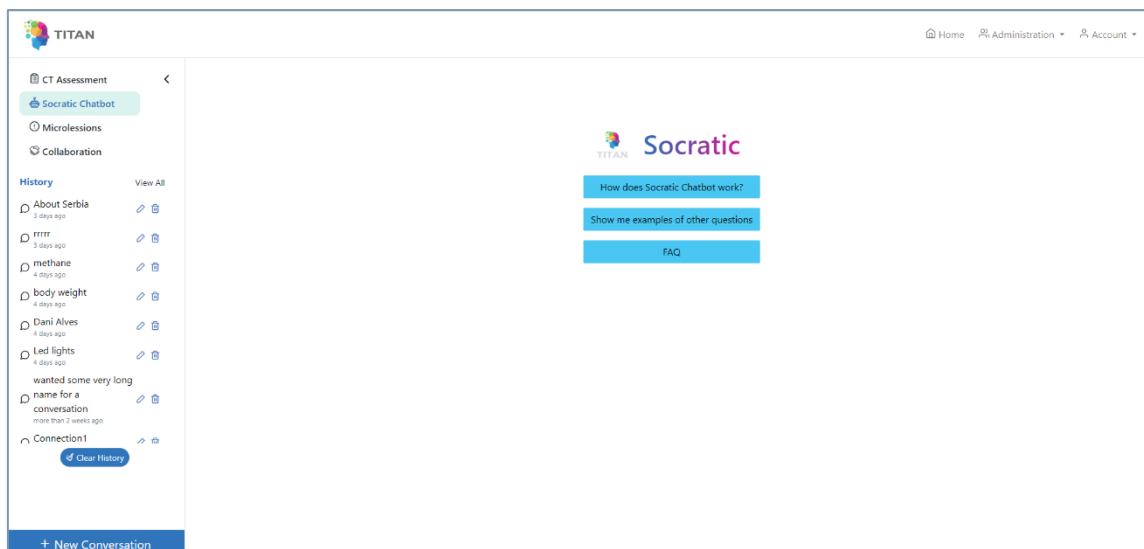
When user clicks on a microlesson name, the modal opens which shows the content of the microlesson to the user (Figure 34).



*Figure 34. Microlesson View.*

### 8.2.3 Socratic Chatbot

The Socratic Chatbot section is presented in 4.4. In the central part of the page, users will find three buttons. The content for these buttons is currently under creation from experts partners. If a user wishes to communicate with the Chatbot, they need to either create a new conversation or select one of the already created conversations from the left side menu.

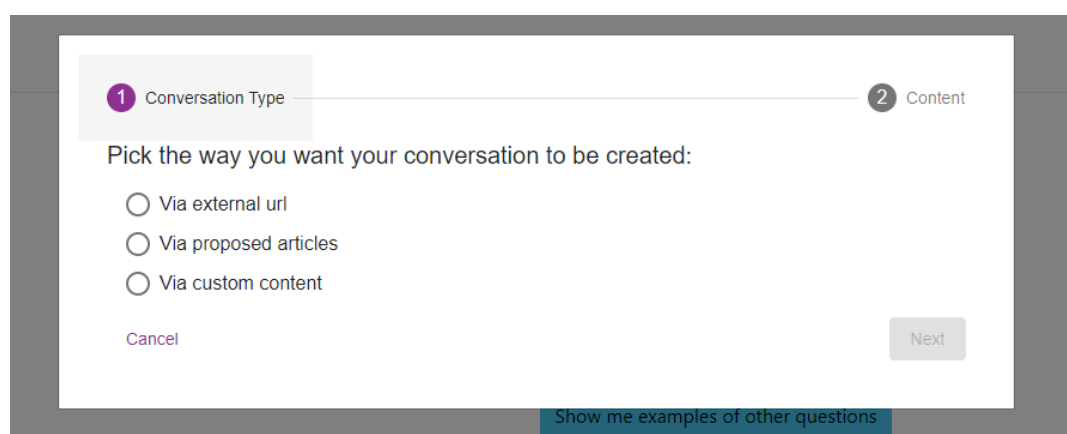


*Figure 35. Socratic Chatbot Section.*

When a user clicks on the "+ New Conversation" button, a modal will open, revealing a stepper form inside (Figure 36). There are two steps for creating a conversation. In the first one, user just needs to choose the desired way of creating a conversation. There are three ways of creating a conversation:

- Via external link
- Via proposed articles
- Via custom content

When user picks one of the proposed ways, the next button in bottom right corner of the modal will be enabled. By clicking on it, user proceeds to the second step. In the second step there will be a form displayed to the user. The kind of form depends on the choice that user has made in the first step.



*Figure 36. New Conversation Creation.*

When this process is completed, user will be prompted to the page where he will be able to send messages to the chatbot inside the conversation he has just created. The conversation name will appear in the left side menu as a history item in the list of all created conversations. The name will be displayed in blue letters to indicate that it is the currently opened conversation. Additionally, users will notice the first message from the Chatbot at the top of the page, which serves as the introduction message (Figure 37).



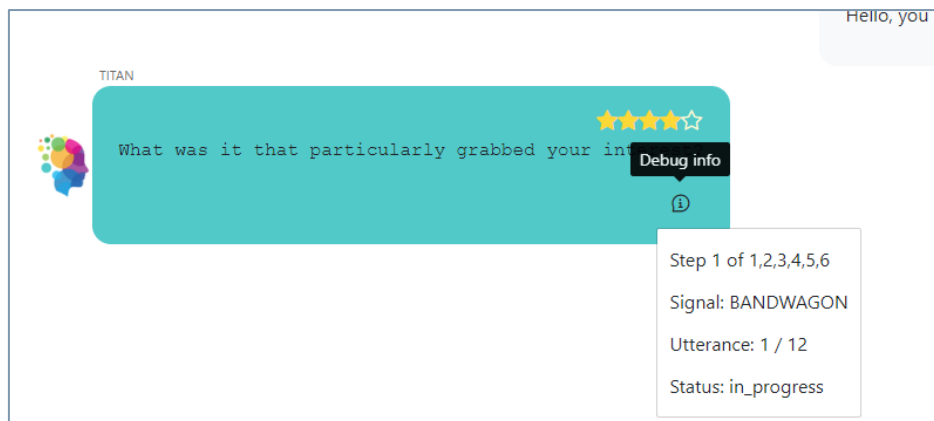
Figure 37. Socratic Conversation start.

If a user wishes to send messages to the Chatbot, they can enter the text inside the input field located at the bottom of the page. Once the user has finished typing the message, they can either press 'Enter' on the keyboard or click on the green button located on the far-right side of the input. Upon sending the message, it will be displayed on the page and will float to the right (Figure 38).



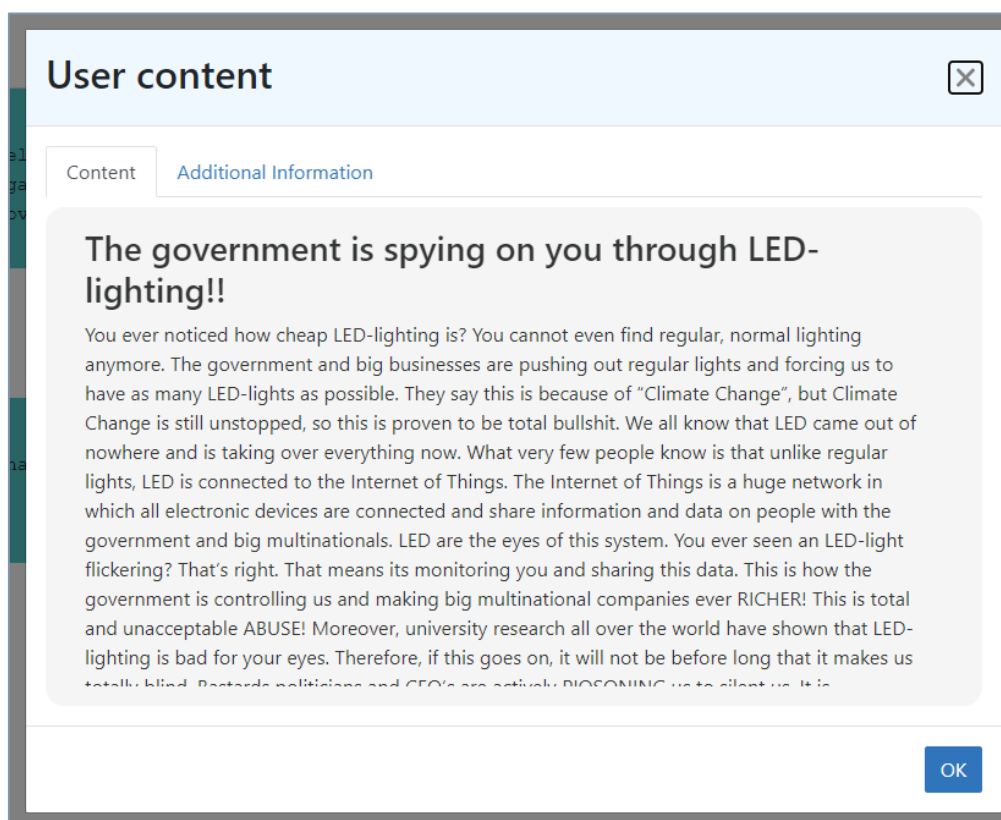
Figure 38. Socratic Conversation continuation.

Every user has the ability to rate the Chatbot's message by clicking on the rating stars located in the top right corner of the message box. Additionally, users can access debug information by clicking on the "debug info" button located in the bottom right corner of the message box (Figure 39).



*Figure 39. Socratic Message details.*

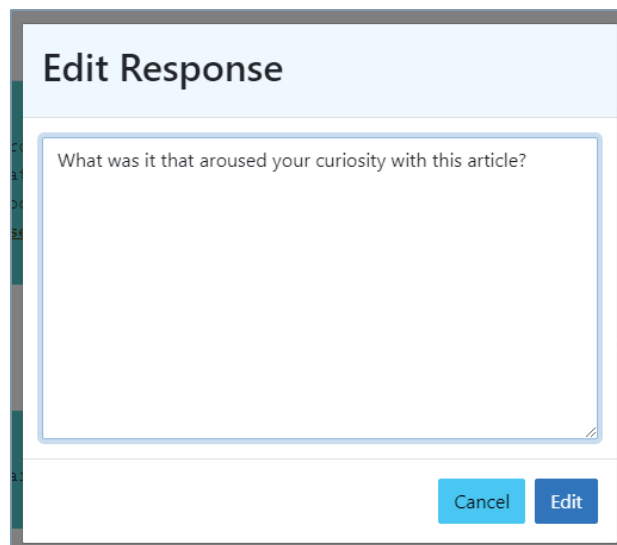
Users can also review the content associated with their previous conversation by clicking on the "See content" button inside the text input bar (Figure 40). Upon clicking this button, a modal will appear where users can read the article corresponding to the conversation, they previously engaged in. At this stage, users can also access additional information about the selected content by clicking on the "Additional Information" tab.



*Figure 40. User Content in Socratic Conversation.*

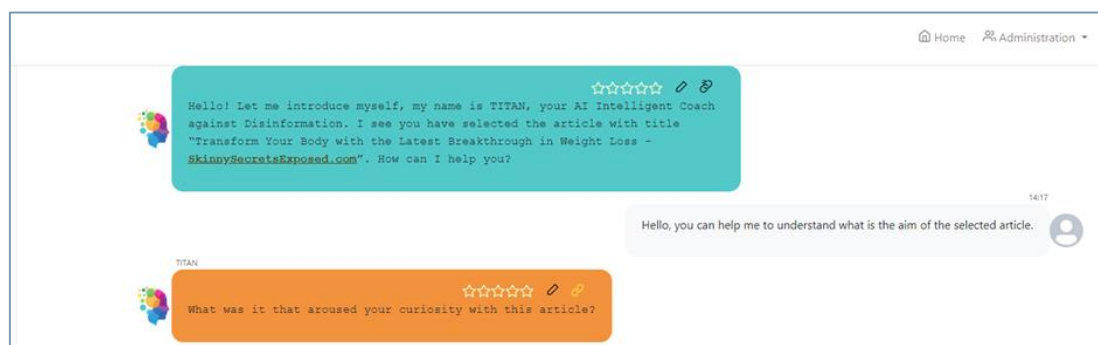
Every user can edit conversation name and delete previously created conversations. In order to do that, he needs to click on buttons with 'pencil' and 'trash-can' icons, respectively from the history menu. Super users can edit the chatbot's response by clicking on the pencil button on the top right of the message box. By clicking on the "Edit" button within the modal, the user can edit the content of the Chatbot's message (Figure 41).





*Figure 41. Edit Chatbot message.*

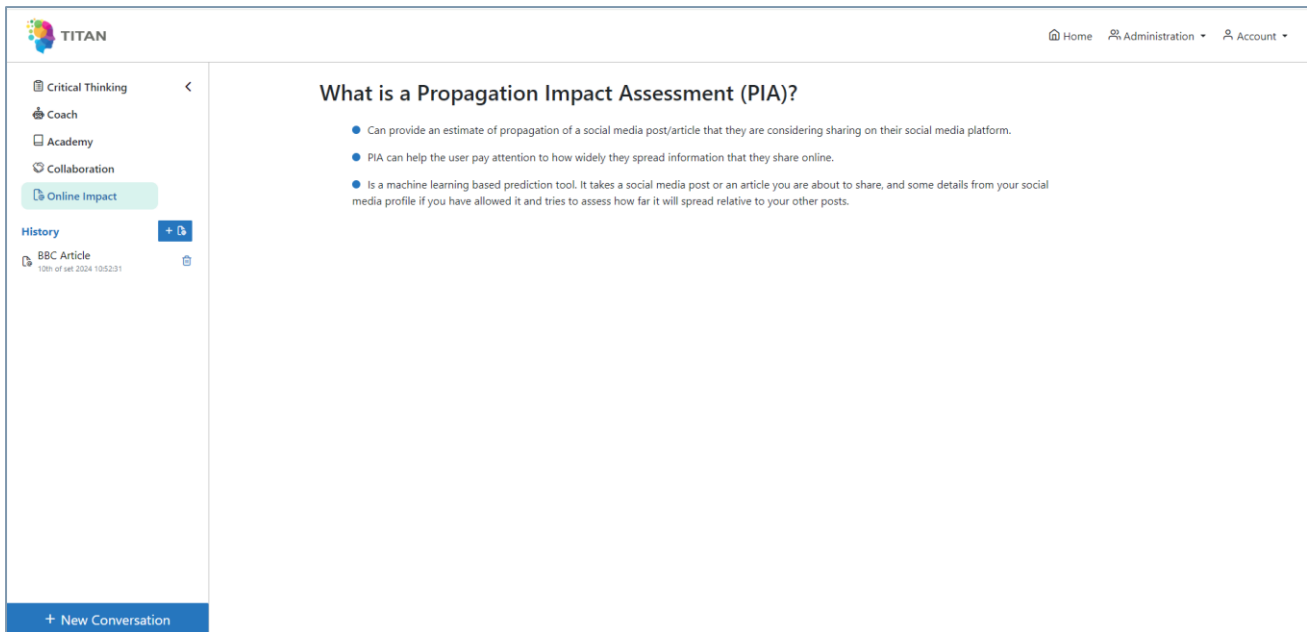
Right next to the "pencil" button, there is an "unlink" button (Figure 42). When pressed, it unlinks the message from the Chatbot in the current conversation. The unlinked message box will turn orange. Additionally, the "unlink" button will transform into a "link" button, which performs the opposite operation of the "unlink" button, allowing the message to be linked back to the Chatbot in the conversation.



*Figure 42. Unlink messages.*

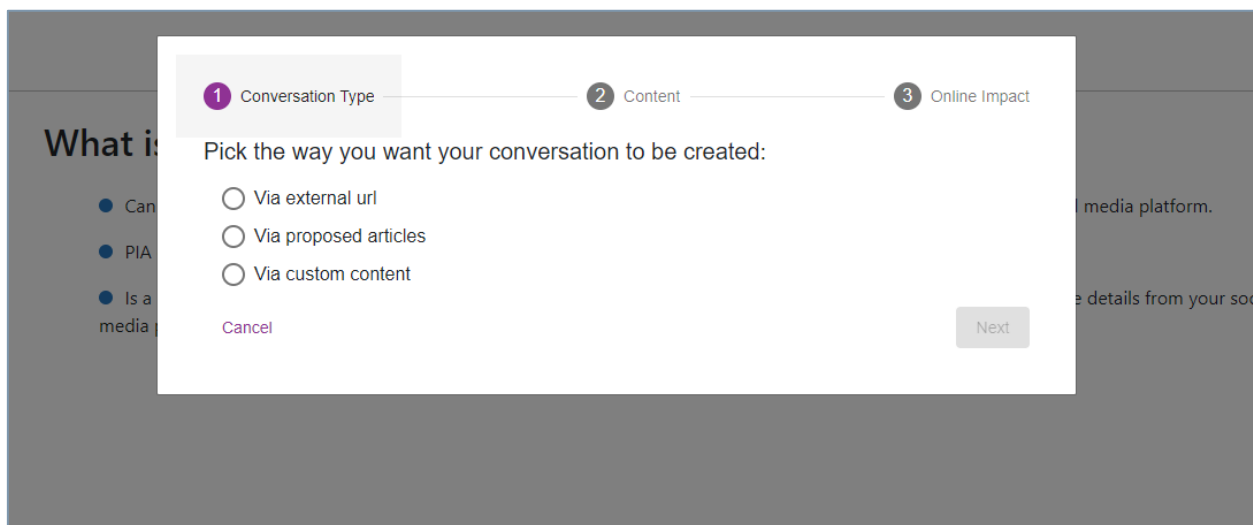
#### 8.2.4 Propagation Impact Assessment (PIA)

The PIA functionality can be accessed from the left-hand side menu, where the initial page shows informative text (see Figure 43). The history of PIA sessions is given on the left side box.



*Figure 43. PIA initial page.*

Upon clicking the “New Online Impact” blue button on top of the sessions history, the user is asked to select the content, with the same modality presented for the chatbot: the user will be able to select a proposed article, an article via URL or free text (see).



*Figure 44. PIA Session creation.*

In addition, the user will select an already registered social media account, or she can add a new one (see Figure 45).

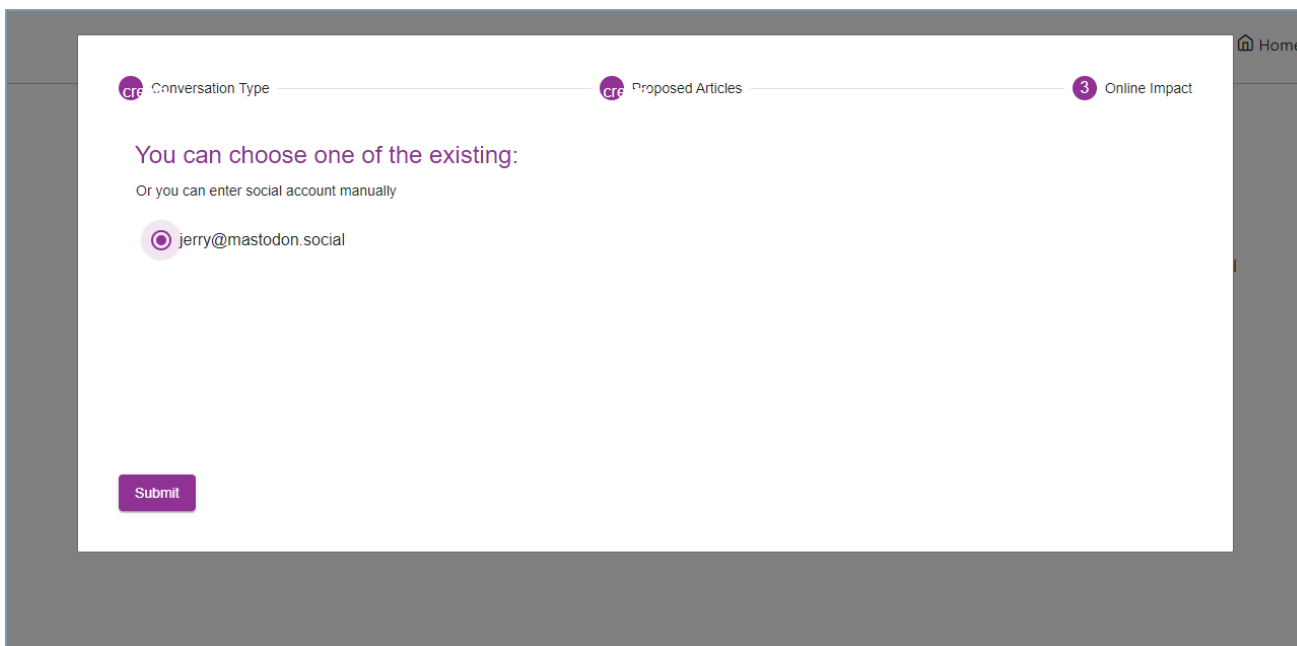


Figure 45. PIA: selection or addition of social media credentials.

After submission, the results (see Figure 46) is presented in form of a *wave image*, in the forms of big and small wave indicating the type of impact. The result page also shows the selected content on the bottom.

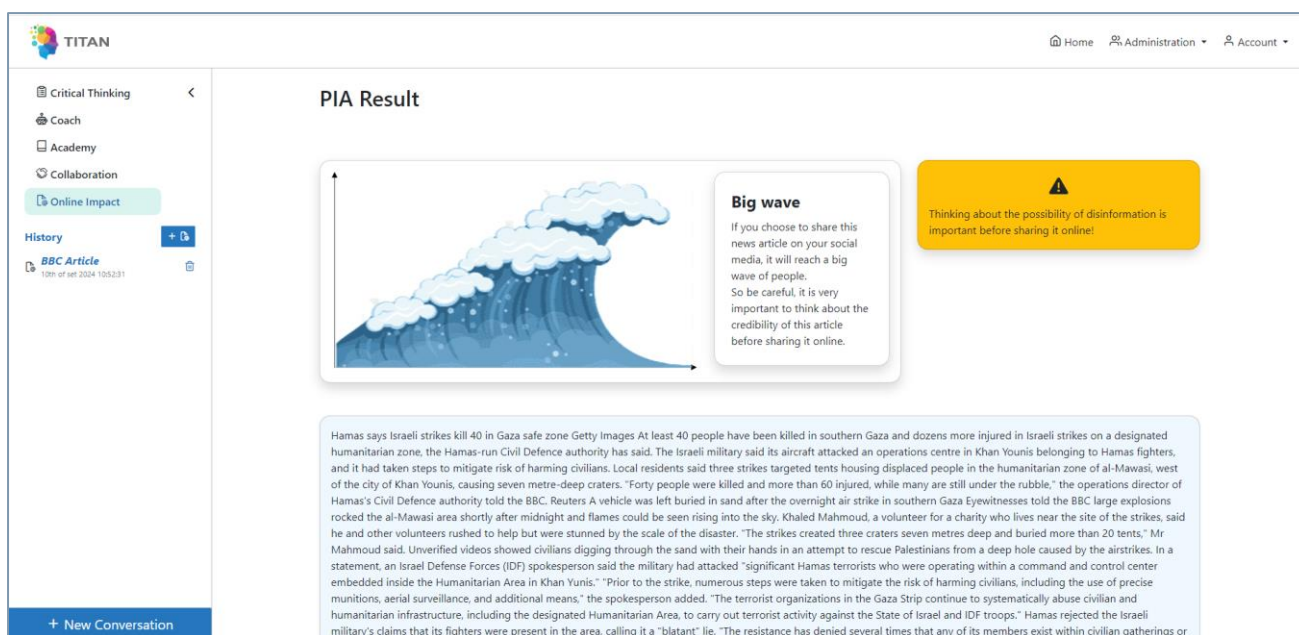
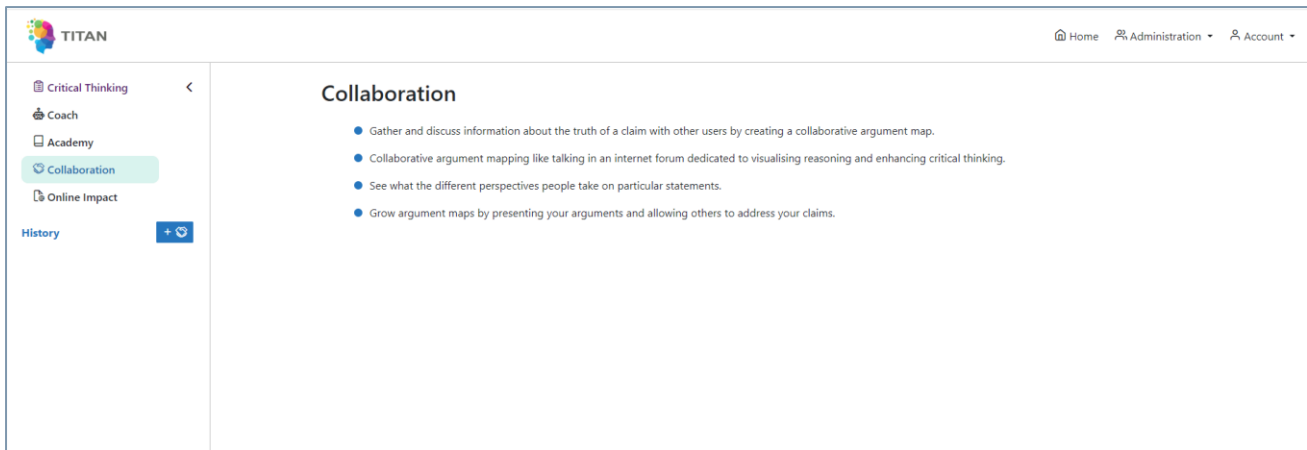


Figure 46. PIA result.

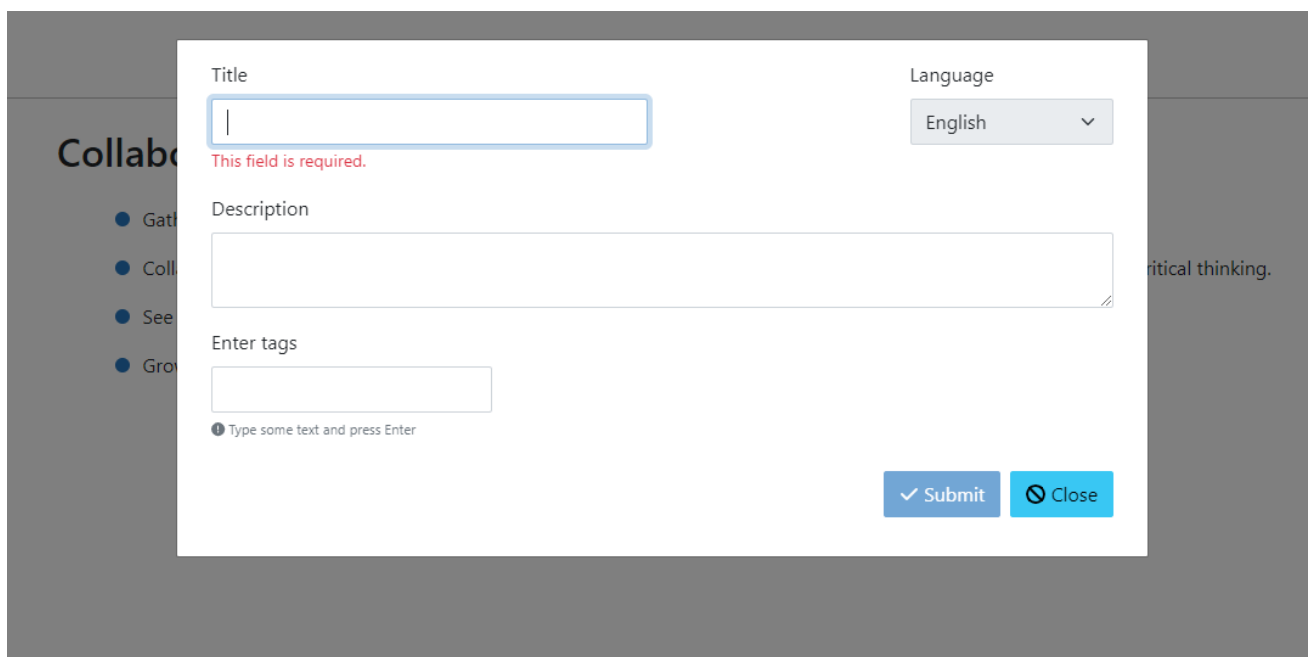
### 8.2.5 Collaboration

The Collaboration functionality can be accessed from the left-hand side menu, where the initial page shows informative text (see Figure 43). The history of collaboration sessions is given on the left side box.



*Figure 47. Collaboration initial page.*

Upon clicking the “New Collaboration” blue button on top of the session’s history, the user is asked to specify the title, description and tags for the session (see Figure 48). The link with the user content to extract initial argumentations is under development.



*Figure 48. Collaboration session creation.*

After submission, the Collaboration environment is show to the user who can now add new argumentations in the canvas (see Figure 49).

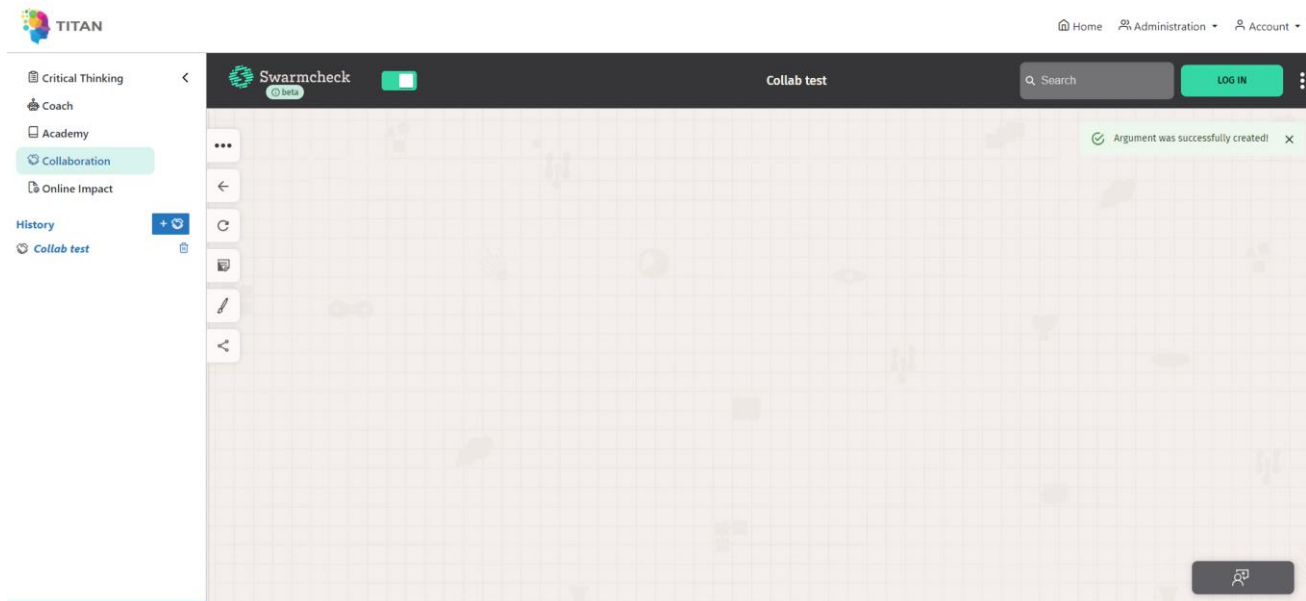


Figure 49. Collaboration session.

## 9 TITAN FRAMEWORK INFRASTRUCTURE

The infrastructure must tackle software, hardware, and network constraints to ensure peak performance and deliver a reliable, resilient, and scalable system. It comprises different layers, each with specific responsibilities.

Figure 50 depicts the general and ideal layer subdivision of an infrastructure<sup>6 7</sup>. The *Software Layer* manages software solutions for business logic, efficient data storage, resource maintenance, and user interaction. It encompasses the *Application Layer* for user interaction, supported by the *Service Layer* for backend functionality. Microservices within the *Presentation Tier* offer REST APIs and messaging channels, while the *Logic Tier* implements business logic. The *System Orchestration Tier* handles incoming requests.

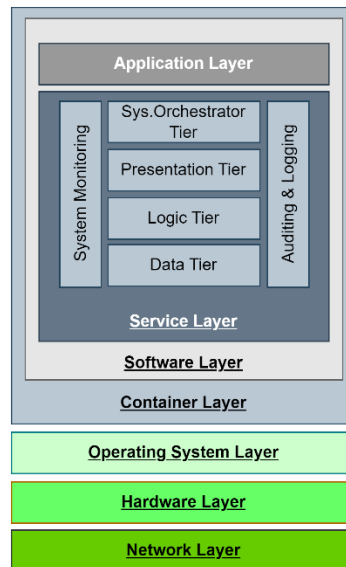
*The Container Layer* packages software into containers for seamless deployment across environments, ensuring proper execution by communicating with the Operating System.

*The Hardware Layer* comprises the system's hardware components, adaptable to different usage scenarios, with specific requirements outlined for the TITAN system.

*Networking* is crucial for efficient system performance, particularly in retrieving information from online sources. HTTP connections facilitate secure communication, mitigating external threats.

<sup>6</sup> NIST Big Data Public Working Group. url: <https://bigdatawg.nist.gov/home.php>

<sup>7</sup> NIST Big Data Interoperability Framework: Volume 6, Reference Architecture. Version 3. url: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-6r2.pdf>



*Figure 50. Infrastructure Layers.*

## 9.1 Software Infrastructure

The software infrastructure incorporates technologies capable of implementing a microservices architecture, which consists of numerous microservice applications, each designed uniquely to handle specific low-level business operations consistently. These microservices are orchestrated by the API Gateway, responsible for managing their calls and outputs to provide high-level business functionalities. Microservice applications register themselves in a Service Registry, allowing the API Gateway to access their endpoints for incoming requests.

The TITAN Architecture comprises multiple microservice applications for business functions, along with transversal layers facilitating communication and system performance monitoring, audit log tracking, and chain of custody maintenance. Each microservice application within TITAN operates as a standalone solution, equipped with technologies tailored to its multi-tier architecture.

The architecture of a microservice application typically consists of three tiers:

- **Data tier:** manages data in one or more database management systems (DBMS) through CRUDL activities.
- **Logic tier:** contains the business logic that integrates data from the Data tier to execute high-level actions meeting user demands.
- **Presentation tier:** provides Application Programming Interfaces (API) for data interchange with other parties and the execution of functionality, such as graphical interfaces and REST APIs.

The TITAN Platform employs a range of technologies to implement its architectural elements, as outlined below:

- **Data Tiers of Microservice Applications:**
  - PostgreSQL: Used as a relational DBMS.
  - MongoDB: Employed as a NoSQL DBMS.
  - Elasticsearch: Utilized as an indexing and search engine.
  - MinIO and OpenIO: Serve as object storage solutions for raw data.
- **Logic Tiers of Microservice Applications:**
  - Spring Boot framework: Enables the development of business functionalities and CRUDL operations on databases.
  - Spring Security with OAUTH2 (Keycloak): Manages authentication and authorization mechanisms.

- Netflix Eureka client: Facilitates the registration of microservice applications into the Service Registry.
- Apache Kafka client: Allows publishing/consuming messages into/from the Messaging layer.
- Presentation Tiers of Microservice Applications:
  - Spring Web: Exposes REST APIs for user interaction.
- API Gateway and Service Registry:
  - Spring Boot framework: For developing business functionalities and CRUDL operations on databases.
  - Spring Security with JSON Web Tokens (JWTs): Handles authentication and authorization mechanisms.
  - Netflix Eureka server: Stores information of registered microservice applications.
  - Spring Cloud Config server: Manages externalized configuration of microservice applications.
  - Hazelcast: Maintains sessions in distributed environments.
  - Netflix Ribbon: Acts as a load balancer, providing client-side load balancing algorithms and failure resiliency mechanisms.
  - Netflix Hystrix: Acts as a circuit breaker, isolating points of access between services, preventing cascading failures, and providing fallback options.
- Messaging Layer:
  - Apache Kafka: Used as a distributed messaging system.
  - Apache Zookeeper: Acts as a centralized service to coordinate Kafka nodes.
- For the System monitoring layer:
  - Dropwizard metrics: Provides insights into Java Virtual Machine status, including active threads, garbage collector activity, and memory usage, as well as metrics for HTTP requests, and active database connections.
  - Swagger: Used for documenting and testing available microservices endpoints.
- For the Auditing and logging layer:
  - Logback: Utilized as a framework to centralize logs from microservice applications and configure logging levels across different system components.
  - ELK Stack (Elasticsearch, Logstash, and Kibana): Facilitates the ingestion, aggregation, and visualization of information extracted from system logs.

## 9.2 Networking

In order to prevent attacks that pose a serious risk to data confidentiality, the TITAN server ports left open are limited, to face the risk of incidents such as stolen information, denial of service, corruption of stored data, etc.

To implement a control mechanism over the server ports, a Nginx<sup>8</sup> instance will be added to the TITAN Platform backend as a reverse proxy, allowing requests to be sent to the backend from any external client, using just one server port. Furthermore, only HTTPs connections are permitted, and the reverse proxy is set up to redirect all HTTP requests to the HTTPs protocol to protect data sent between a client and the TITAN Platform backend and prevent sniffing or spoofing.

To verify the legitimacy and dependability of the communications to third parties, a certificate issued by a certificate authority (such as Let's Encrypt<sup>9</sup>) will be placed on the server side when the TITAN Platform is deployed.

A firewall may be required to control port opening, shutting, forwarding, and denying rules as well as filter network traffic that the system exchanges with the external network to enhance security. A host-based firewall, or software firewall that is installed on a machine, is sufficient for a single-node server.

---

<sup>8</sup> <https://www.nginx.com/>

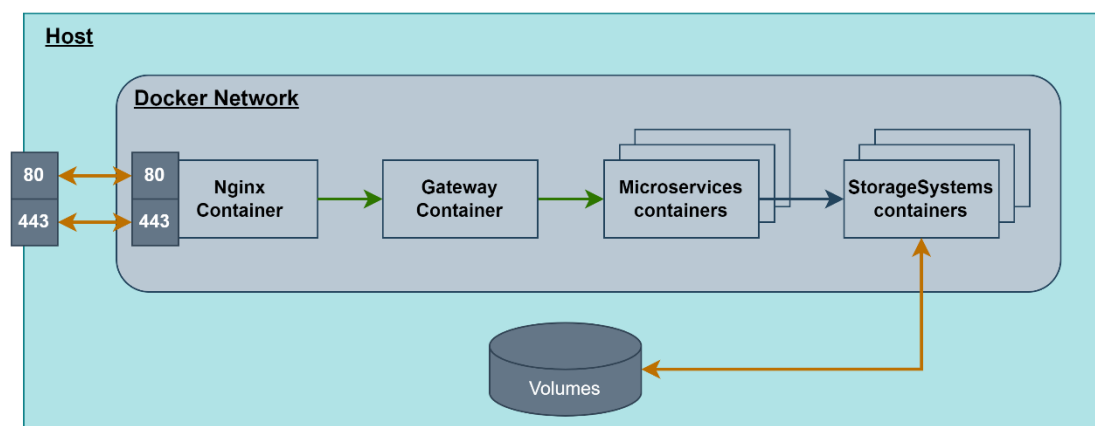
<sup>9</sup> <https://letsencrypt.org/>

## 10 CONTINUOUS INTEGRATION AND DELIVERY

To support the iterative development model and ensure that the process is heading in the right direction, TITAN adopts a Continuous Integration and Delivery (CI/CD) approach. This strategy aims to streamline and accelerate the software lifecycle while facilitating a fast feedback loop. Combined with a well-planned schedule of release reviews during living labs and pilots, this approach helps in gathering and incorporating early feedback into future releases as quickly as possible.

Every project solution will be delivered as a Docker image, as the consortium agreed. Because of its widespread use, cross-platform compatibility, ease of installation, relocation, and maintenance, as well as its isolation and container independence, Docker has been chosen as the container technology.

A Docker Compose file with the definition of containers to instantiate, environment variables, resources to allow for each container, and volumes to store data will be provided to ease the deployment of TITAN Platform containers (Figure 51). The ports of the Nginx reverse proxy will be bound only to the ports of the hosting machine in order to maintain the isolation of the solution and restrict external access while still enabling online access to the TITAN system.



*Figure 51. Deployment via Docker Compose.*

An instance of Gitlab<sup>10</sup> has been deployed on the TITAN server and it will be used to manage and automatize the deployment activities.

<sup>10</sup> <https://about.gitlab.com/>



## 11 CONCLUSIONS

This deliverable reports about the design and implementation of the first major release and subsequent minor releases of the TITAN platform and its underlying services.

The first release of TITAN builds upon the socio-technical theoretical background and user needs elaborated in WP2, shaped and refined through co-creational workshops and Living Labs sessions conducted in WP3, and through an iterative development approach aiming to incorporate early feedback through review sessions conducted in WP3 and WP5 to ensure that the development process is heading in the right direction.

To support this iterative development model and the incorporation of fast feedback loop, TITAN adopts a Continuous Integration and Delivery (CI/CD) approach.

To properly and effectively transfer the theoretical foundation, the user needs, and early feedback into product features and technical specifications, TITAN has applied an iterative requirement management process, fostering continuous interdisciplinary collaboration and ensuring that requirements reflect the actual needs of the target audience. To support the consortium in securing ethics approvals, especially for human participation and proper AI management, an External Ethics Advisory Board (EEAB) was established. The iterative requirement management process incorporates the EEAB's recommendations, and full compliance details will be included in the second version of the Report on Legal and Ethical Impact Assessment (D1.4).

The platform provides various functionalities aimed at empowering citizens in combating disinformation and misinformation. Among these features, in the first release of TITAN the rule-based implementation of the Socratic Chatbot stands out, designed to engage users in meaningful conversations and encourage critical reflection on their beliefs. Socratic dialogues are customized based on disinformation signals detected within user-supplied textual content and on their critical thinking abilities, assessed through platform-administered questionnaires.

The first version of the Socratic Chatbot is based on a rule-based conversational model to ensure a structured and predictable conversational flow. This approach was deemed crucial for gathering insights on usability, impact, acceptance, and attitude during the initial stages of Living Labs and the first Pilot iteration.

Additionally, as a consequence of the iterative development and continuous integration process, the first release of TITAN incorporates two pivotal modules to assist users in understanding other essential aspects of information management that were originally scheduled for the second release but instead have been integrated in the first release to start collecting early feedback. Specifically, these are the social media impact related to content sharing and the argumentative analysis of the content itself. These functionalities are covered by the Propagation Impact Assessment and Collaboration Environment modules, respectively.

Furthermore, the collection and curation of diverse datasets lie at the core of TITAN, serving as a pivotal foundation for refining the AI models integral to their functioning. Our research journey focused on identifying and crafting datasets encompassing various disinformation signals, including hate speech, offensive language, toxic language, clickbait, and logical fallacies. Additionally, our work extends to argumentation mining, with structured datasets tailored for effective processing and analysis. The subsequent pre-processing phase involved rigorous data cleaning and duplicate removal to ensure the datasets' quality, consistency, and accuracy. These refined datasets were then utilized to fine-tune Transformer-based Language models capable of adeptly detecting hate speech, offensive language, and clickbait in English text, providing a robust automated system for identifying disinformation.

In summary, the first implementation of TITAN was primarily focused on gathering early and effective feedback during the Living Labs and pilots through an iterative development approach. The aim was to ensure that the development process stayed on the right track. While some of this feedback and insights were promptly addressed, fixed, and integrated into subsequent minor releases, the most significant ones have been taken into account to guide the implementation of the second release, expected by M30.

## 12 REFERENCES

- A. Conneau *et al.*, ‘Unsupervised Cross-lingual Representation Learning at Scale’, *CoRR*, vol. abs/1911.02116, 2019.
- Y. Liu *et al.*, ‘RoBERTa: A Robustly Optimized BERT Pretraining Approach’, *CoRR*, vol. abs/1907.11692, 2019.
- P. He, J. Gao, and W. Chen, ‘DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing’, *CoRR*, vol. abs/2111.09543, 2021. <http://arxiv.org/abs/2111.09543>.
- T. Davidson, D. Warmesley, M. Macy, and I. Weber, “Automated Hate Speech Detection and the Problem of Offensive Language”, *ICWSM*, vol. 11, no. 1, pp. 512-515, May 2017.
- J. Salminen, H. Almerexhi, M. Milenković, S.-gyo Jung, J., An, H. Kwak and B. Jansen (2018). Anatomy of Online Hate: Developing a Taxonomy and Machine Learning Models for Identifying and Classifying Hate in Online News Media. *Proceedings of the International AAAI Conference on Web and Social Media*, 12(1). <https://doi.org/10.1609/icwsm.v12i1.15028>
- J. Qian, A. Bethke, Y. Liu, E. Belding, and W. Y. Wang, ‘A Benchmark Dataset for Learning to Intervene in Online Hate Speech’, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 4755–4764.
- I. Mollas, Z. Chrysopoulou, S. Karlos, and G. Tsoumakas, ‘ETHOS: a multi-label hate speech detection dataset’, *Complex & Intelligent Systems*, vol. 8, no. 6, pp. 4663–4678, Dec. 2022.
- L. Grimminger and R. Klinger (2021, April). Hate Towards the Political Opponent: A Twitter Corpus Study of the 2020 US Elections on the Basis of Offensive Speech and Stance Detection. In O. De Clercq, A. Balahur, J. Sedoc, V. Barriere, S. Tafreshi, S. Buechel, & V. Hoste (Eds.), *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 171–180). Retrieved from <https://aclanthology.org/2021.wassa-1.18>
- M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, ‘Predicting the Type and Target of Offensive Posts in Social Media’, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 1415–1420.
- A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, ‘Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media’, *CoRR*, vol. abs/1610.09786, 2016.
- Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast, ‘Webis Clickbait Spoiling Corpus 2022’. Zenodo. doi: 10.5281/zenodo.6362726.  
<http://arxiv.org/abs/1907.11692>

### 13 ANNEX 1: SOCRATIC DIALOGUE FLOWCHARTS

