

AI-CODE

AI services for COntinuous trust in emerging Digital Enviroments

D4.1

PROTOTYPING THE DISINFORMATION DETECTION AND NEXT GENERATION OF SOCIAL MEDIA SERVICES

Grant Agreement no.	101135437
Project Title	AI-CODE - AI services for Continuous trust in emerging Digital Environments
Contractual Due date	31/05/2025
Actual submission date:	11/06/2025
Lead beneficiary:	DS TECH

Deliverable number	D4.1
Deliverable Title:	Prototyping the disinformation detection and next generation of social media services
Description:	The deliverable presents the initial versions of the Disinformation detection services focusing on next generation social media, as carried out in T4.1.
Type/Nature	☐ R - <i>Document, report</i> ☒ DEM - <i>Demonstrator, pilot, prototype</i> ☐ DEC - <i>Websites, patent filings, videos, etc.</i> ☐ OTHER ☐ ETHICS – <i>Ethics requirement</i> ☐ ORDP – <i>Open Research Data Pilot</i>
Dissemination Level	☒ Public – fully open ☐ Sensitive — limited under the conditions of the Grant Agreement ☐ EU classified — RESTREINT-UE/EU-RESTRICTED, CONFIDENTIEL-UE/EU-CONFIDENTIAL, SECRET-UE/EU-SECRET under Decision 2015/444. For items classified under other rules (e.g. national or international organization), please select the equivalent EU classification level.
Work Package number	WP4
Work Package leader	FBK
Author(s) <i>in alphabetical order</i>	Beatriz Arregui Garcia, Lucio La Cava, Róbert Móro, Giovanni Natale, Giulio Spinozzi, Andrea Tagarelli
Internal Reviewers <i>in alphabetical order</i>	Ilias Kanellos (ATC)

AUTHORS

Name	Partner	E-Mail
Beatriz Arregui Garcia	FBK	barreguigarcia@fbk.eu
Lucio La Cava	UNICAL	lucio.lacava@dimes.unical.it
Róbert Móra	KInIT	robert.moro@kinit.sk
Giovanni Natale	DS TECH	g.natale@dstech.it
Giulio Spinozzi	DS TECH	g.spinozzi@dstech.it
Andrea Tagarelli	UNICAL	andrea.tagarelli@unical.it

HISTORY OF CHANGES

Version	Date	Changes
0.1	15/04/2025	Table of Contents
0.2	05/05/2025	First version
0.3	19/05/2025	Final draft for internal review
0.4	26/05/2025	Comments and suggestions by internal reviewers
0.5	31/05/2025	Integration of feedback from internal reviewers
0.6	03/06/2025	Review by lead beneficiary (DS TECH)
1.0	10/06/2025	Final version prepared

TABLE OF CONTENTS

1. Executive Summary	6
2. Introduction	6
2.1 Goal and Motivation	7
2.1.1 Fediverse Network Moderation Tool	7
2.1.2 Mastodon Instance Inspector	8
2.2 Scope of the Prototype	8
2.2.1 Connection to the requirements defined in D3.1	9
2.2.2 Fediverse Network Moderation Tool	10
2.2.3 Mastodon Instance Inspector	11
3. Scientific Rationale	12
4. System Description	15
4.1 Prototype Architecture	15
4.2 Technical Description	17
4.2.1 Usage and Access Information	18
4.3 Ethical Design Considerations	25
5. Next Steps	25
6. Conclusions	27
7. Annex: Deployment and Configuration Details for AI-CODE Services	27
7.1 A.1 Initial Configuration	28
7.2 A.2 Mastodon_NetMod Service	28
7.2.1 A.2.1 Dockerfile	28
7.3 Stage 1: Build	28
7.4 Stage 2: Runtime	28
7.4.1 A.2.2 Docker Compose Configuration	28
7.5 A.3 Mastodon Instance Inspector Service	29
7.5.1 A.3.1 Dockerfile	29
7.6 Stage 1: Build	29

7.7	Stage 2: Runtime	29
7.7.1	A.3.2 Docker Compose Configuration	30
7.8	A.4 Service Exposure via Traefik	30
7.9	A.5 Service Startup	30
7.10	A.6 Planned Enhancements	31

Table of figures

Figure 1	Prototype-Docker Single Node Architecture	17
Figure 2	Prototype - Docker Multi-Node Architecture	17
Figure 3	Example Endpoint NetMod	18
Figure 4	Sample executed Operation NetMod	19
Figure 5	Mastodon Admin Panel for moderating a single instance	20
Figure 6	Mastodon Admin Panel for importing moderation strategies from other instances	21
Figure 7	Example endpoint Instance Inspector	21
Figure 8	MTG Score vs Degree Example graphical	22
Figure 9	MGT Probability Distribution Graph	23
Figure 10	Check-worthyness Probability Distribution	23
Figure 11	Top Hashtags	24

Table of tables

Table 1	Functional requirements as seen in D3.1	10
Table 2	Non-Functional requirements as seen in D3.1	10
Table 3	Ethical considerations.	25

1.Executive Summary

This deliverable introduces the prototype of a disinformation detection service for federated social media, developed under Task 4.1 of Work Package 4 (WP4) in the AI-CODE project. The service integrates two tools—the Fediverse Network Moderation Tool and the Mastodon Instance Inspector—to support journalists, fact-checkers, and researchers in assessing moderation behavior and disinformation risks across Mastodon instances.

In this first prototyping phase, the applications are developed according to the requirements collected in the previous deliverables, therefore developed through docker containers but without a graphical interface for the moment, therefore accessible from the terminal and through a proxy.

2.Introduction

This deliverable describes a prototype of the *Disinformation Detection in the Next-Generation Media* service, which is a result of task T4.1. The service combines two tools:

- **Fediverse Network Moderation Tool**, the main aim of which is to support administrators in maintaining safe, inclusive online spaces in the next-generation (federated) social media by streamlining moderation tasks and fostering cross-instance collaboration.
- **Mastodon Instance Inspector**, which aims to provide valuable insights into the structure, activity, and health of Mastodon instances, enabling informed decision-making and better network management.

Prototypes of both tools have been developed and deployed and are available online:

- Fediverse Network Moderation Tool: [Usage and Access Information](#)
- Mastodon Instance Inspector: [Usage and Access Information](#)

The deliverable is structured as follows:

- The remaining part of **Section 2** introduces the goal and motivation behind both tools, which together comprise the *Disinformation Detection in the Next-Generation Social Media* service. It also defines the scope of the developed prototype, namely connection to relevant requirements and specifications defined in deliverables D3.1 and D3.2 and the functionalities and use cases addressed by the prototype.
- In **Section 3**, we summarize the scientific rationale behind the service as introduced in more details in deliverables D2.1 and D2.2. We also mention relevant technical recommendations that influenced the design of the service as proposed in D2.3.
- In **Section 4**, we describe the prototype in detail – its architecture and important technical details as well as supported inputs and outputs of the prototype. In addition, we discuss how the ethical requirements are addressed by the current prototype.
- In **Section 5**, we provide an outlook on further development of the service as a part of WP5 as well as its evaluation in WP8.

2.1 Goal and Motivation

The decentralized nature of the Fediverse, and Mastodon in particular, introduces both opportunities and challenges in moderating content and ensuring trustworthy engagement. Unlike centralized platforms, Mastodon instances operate independently, each with their own rules, moderation practices, and community standards. While this enables community autonomy, it also complicates efforts to assess the credibility and safety of instances, and to detect coordinated disinformation.

This decentralized architecture makes it difficult for journalists, researchers, moderators, and policymakers to maintain situational awareness across the network. Inconsistent moderation visibility, fragmented data on bans and content, and the lack of a centralized reputation system create blind spots that can be exploited by disinformation actors.

The services prototyped in this task aim to address these issues by enabling transparency, traceability, and comparative analysis of instance behaviours and content signals across the Fediverse. By identifying harmful moderation patterns, surfacing disinformation signals, and analysing information flow, these tools empower professionals and communities to make informed decisions about trust and safety in the next generation of social platforms.

This approach directly supports AI-CODE's broader mission to foster continuous trust in digital environments through AI-driven, user-centred services.

2.1.1 Fediverse Network Moderation Tool

The goal of the **Fediverse Network Moderation Tool** is to help the discovery of potentially harmful servers in the Fediverse, keeping the pace with the fast-evolving nature of decentralized social media. Indeed, in a scenario where decentralization requires distributed yet manually enforced moderation strategies, this service aggregates and categorizes banned servers across specific keywords (e.g., spam, harassment, racism, etc), and tracks how frequently a server has been banned under each category, thus making it easier for both experts and non-experts in the field to access these information and improve instances' interactions within the Fediverse.

The prototype developed in this phase implements the key functionalities of the **Fediverse Network Moderation Tool**. It currently enables the periodical fetching (download data and instance information and statistics) of online Mastodon instances, from which it collects moderation strategies by means of the official Mastodon APIs. These are thus made available to end-users, who can exploit them to investigate the moderation dynamics within Mastodon and the broader Fediverse.

The scenario of use addressed by the prototype aligns with the one presented in Deliverable D3.1. For example:

- Journalists: can prevent the spreading of harmful content;
- Server admins and moderators: can proactively block harmful instances before they interact with their community;

- Governments: can identify patterns of misinformation spread on decentralized social networks and proactively act;
- Policymakers: can follow the moderation dynamics to shape new policies.

2.1.2 Mastodon Instance Inspector

The goal of the **Mastodon Instances Inspector** service is to facilitate and improve the ability of journalists, fact-checkers, and communication professionals to verify the credibility and moderation behaviour of Mastodon instances within the decentralized Fediverse. In a landscape where misinformation can propagate rapidly and traditional moderation frameworks are decentralized, this service provides an infrastructure for transparency, trust evaluation, and responsible engagement.

The prototype developed in this phase focuses on delivering the foundational functionality of the Mastodon Instances Inspector. At this stage, it enables users to investigate the moderation history of Mastodon instances, particularly by analysing bans issued and received, and understanding how instances are positioned within the broader network of moderation practices. Additionally, the prototype offers experimental support for AI-assisted content analysis, currently limited to textual data from a single instance (mastodon.social). Ultimately, the service aims to support the generation of visual and data-driven reports, helping users draw meaningful insights into moderation behaviour, content trustworthiness, and the spread of disinformation.

The scenario of use addressed by the prototype aligns with the one presented in Deliverable D3.1. For example:

- Journalists: who benefit from tools that support credible reporting.
- Fact-checkers: who require detailed, traceable evidence of content credibility and moderation patterns.
- PR and Marketing professionals: who need to ensure brand-safe engagement with Mastodon communities.
- Researchers: who study network behaviors and online moderation dynamics.

By providing useful information and clear data, the prototype helps these user groups make better decisions when using the Fediverse and reduces the risks that come with decentralized social networks.

In summary, both tools are designed to reduce the opacity of decentralized moderation, enabling more informed, responsive, and trustworthy engagement with the Fediverse. The following section outlines the current capabilities of each tool.

2.2 Scope of the Prototype

The prototype developed in this phase of the project responds to the challenges identified in Section 2.1 by offering two interrelated tools that enhance transparency and trust assessment across the Fediverse:

- The Fediverse Network Moderation Tool:

- Collects moderation dynamic data from publicly available Mastodon instance endpoints.
- Stores and organizes these data in a queryable format for later retrieval and analysis.
- The Mastodon Instance Inspector:
 - Generates interactive network visualizations to explore moderation relationships between instances.
 - Provides content-level analysis for selected instances, including bot-likelihood, hate speech signals, trending topics, and fact-checking relevance.

Together, these services offer a foundation for understanding moderation practices and detecting disinformation risks within decentralized platforms like Mastodon. The prototype focuses on implementing essential functionalities identified in the early design phases (Deliverables D3.1 and D3.2), paving the way for future expansion in WP5.

While not yet offering a graphical interface, both services are accessible through a command-line interface or proxy endpoint and are designed for extensibility and integration with broader AI-CODE systems.

2.2.1 Connection to the requirements defined in D3.1

The two tables below present the requirements for this service using the same format as in Deliverable 3.1.

#	Early high-level functional requirements	How will the requirement be addressed in the service design?
1	The system shall allow users to input keywords to identify and curate a list of servers to moderate based on specific harmful interactions (e.g., racism, harassment).	Users will be able to input custom keywords (e.g., “racism”, “harassment”, “misinformation”) to identify instances associated with specific harmful interactions.
2	The system should be able to regularly (e.g., daily) crawl the Fediverse for getting the most up-to-date network moderation information from existing servers, that will be used to build the lists	The system will regularly crawl publicly available metadata from Mastodon and other ActivityPub-compatible instances to maintain an up-to-date view of moderation practices and inter-instance bans.
3	The system shall include a database for storing and updating the metadata about network moderation publicly available from each instance	The system will include a database for managing the metadata about network moderation
4	The system shall generate and provide Mastodon-compatible files for enforcing moderation on identified servers.	The system will allow the users to export block lists that adhere Mastodon format.
5	The system shall display historical data on server flags, including the number of times servers have been flagged for specific reasons.	The systems will provide the users with historical data on banned instances.
6	The system shall implement AI-driven content analysis to assess and report on recent posts from instances flagged for misinformation or hate speech.	Leveraging AI-driven models, the system will analyze recent user-generated content (on-demand) from flagged instances, detecting and reporting indicators of misinformation, hate speech, or machine-generated content.
7	The system shall enable users to search for specific instances and retrieve their credibility status within the banning network.	Users will be able to search for any instance and retrieve a trust summary based on its

		position in the banning network, content quality metrics, and moderation history.
8	The system shall provide tools for tracking the spread of content across the Fediverse, identifying key sources involved in misinformation dissemination.	The system will track how content — particularly misinformation or high-engagement posts — spreads across the network, assessing the infodemic risk.
9	The system shall generate visual reports that highlight trends and patterns in information spread and moderation dynamics between instances.	Interactive visualizations will highlight moderation dynamics, ban networks, and content spread.
10	The system shall allow users to flag discrepancies in ban evaluations across different instances and provide summaries of bans produced or received by an instance.	Users will be able to highlight inconsistencies in ban behaviors (e.g., divergent treatment of similar instances) and view summaries of moderation actions performed <i>by</i> and <i>against</i> each instance.
11	The system shall implement network analysis features to quantify moderation participation and polarization among instances based on their banning history.	Built-in graph analysis tools will quantify moderation engagement and centrality enabling deeper insights into governance patterns across the decentralized network.

Table 1 Functional requirements as seen in D3.1.

#	Early high-level non-functional requirements	How will the requirement be addressed in the service design?
1	The application shall be accessible and useable for non-technical audiences	The user interface will be designed with accessibility in mind, offering clear visual cues, simplified workflows, and glossary support to ensure usability by non-technical audiences.
2	The application shall be well documented and described	Documentation shall be provided, including usage instructions, methodology explanations, and glossary terms, to support onboarding, transparency, and reproducibility.
3	The application shall be responsive, with low computation time	Backend services will be optimized for performance to ensure minimal latency and fast response times.
4	The system shall provide a user-friendly interface that facilitates easy navigation and interpretation of data for non-technical audiences.	The interface shall prioritize intuitive navigation and legibility, using clean visual design, contextual tooltips, and guided workflows to simplify interaction with complex data.
5	The system shall ensure scalability to accommodate a growing number of instances and users without performance degradation.	The system architecture will support increasing traffic, data volume, and user load.

Table 2 Non-Functional requirements as seen in D3.1.

2.2.2 Fediverse Network Moderation Tool

This prototype represents the first version of the Fediverse Network Moderation Tool service. It currently focuses on the collection, organization, and storage of moderation dynamics across the Fediverse, with particular emphasis on keeping information up to date given the fast-evolving pace of these platforms. In this regard, it covers several core requirements outlined in D3.1 Section 5.5 (Early Requirements) and detailed further in D3.2 Section 3.1.1 (Functional Specifications and Technical Requirements). Based on these, the next phases of development will focus on extending functionalities to cover the other requirements.

Prototype functionalities:

The Fediverse Network Moderation Tool currently implements two main modules:

1. The Moderation Dynamics Collector
2. A storing mechanism for efficient retrieval of collected dynamics based on user needs.

The first module runs in background on a periodical schedule on a dedicated instance in cloud managed by DST (e.g., daily, weekly), and discovers currently active Mastodon instances in the Fediverse. For each discovered instance, then, it collects the currently enforced moderation strategies with associated metadata (e.g., current date, reason). This module helps having a constantly updated perspective over what are the ban phenomena currently enforced. This prototype leverages commonly adopted Python libraries (e.g., requests) to interact with the official Mastodon APIs.

The second main module is in charge of organizing and storing the collected data in a way that is easily accessible by end-users. It is built using MongoDB and allows users to declare a set of keywords based on which it has to retrieve moderation dynamics. This module is also responsible for properly presenting users with the requested information, so as to foster their usage.

Implemented in the current prototype:

- Scheduled crawling of moderation dynamics from the Fediverse: the service periodically fetches moderation strategies enforced by currently online instances (Covers D3.1 Requirements #2; D3.2 Functionalities #3.1)
- Proper storing of metadata related to moderation dynamics for subsequent usage in the service (Covers D3.1 Requirements #3; D3.2 Functionalities #3.1)

2.2.3 Mastodon Instance Inspector

This prototype delivers the first version of the Mastodon Instances Inspector service. Its current scope focuses on moderation transparency and early-stage content inspection, addressing several core requirements outlined in *D3.1 Section 5.5 (Early Requirements)* and detailed further in *D3.2 Section 3.1.1 (Functional Specifications and Technical Requirements)*. It sets the foundation for extended functionality to be implemented in the next phases of development.

Prototype functionalities:

The Mastodon Instances Inspector currently offers two main analytical modules:

1. Instance and Network Exploration

This module helps users explore the structure of the Fediverse through an interactive network map of Mastodon instances. The visualization highlights how instances are connected—either by federation relationships or moderation actions such as bans. Each node in the graph represents an instance, with edges indicating who bans or follows whom.

The prototype uses interactive network graphs (built with *Cosmograph* and *NetworkX*) to represent these connections. Node size reflects network centrality; colour indicates properties such as moderation behaviour or ban frequency; and edge directionality shows moderation or federation flow. These visual cues help users identify clusters, key hubs, or isolated and controversial instances.

Clicking on a node enables deeper inspection of the selected instance. Although this is currently available only for mastodon.social, it lays the groundwork for broader comparisons and richer interactivity in future releases.

2. Content Analysis of Specific Instances

The second core module focuses on analysing content and user behaviour within individual Mastodon instances. While currently limited to mastodon.social, the system is designed to scale across multiple instances.

Key analytical features include:

- **Bot-likelihood scores:** Posts are evaluated for probability of being machine-generated.
- **Hate-speech signals:** Detection of potentially harmful or abusive content.
- **Posting activity trends:** Aggregated over time to detect spikes or unusual activity.
- **Top active accounts:** Ranked by volume and bot-likelihood.
- **Language distribution:** Breakdown of languages used in the instance.
- **Content safety:** Proportion of NSFW-tagged content.
- **Fact-checking signals:** Check-worthiness scoring and claim extraction for external verification.
- **Trending hashtags:** Identification of emerging topics.

All outputs are visualized through intuitive charts and graphs, helping users identify important patterns without requiring technical expertise. The following sections distinguish between the functionalities already implemented and those planned for upcoming versions.

Implemented in the current prototype:

- *Instance Lookup and Moderation History:* users can search for a Mastodon instance and retrieve an overview of its moderation activity, including bans it has issued or received and the reasons behind those actions. (Covers D3.1 Requirements #3 and #10; D3.2 Functionalities #7 and #10)
- *Initial AI-Assisted Content Analysis (One Instance):* content analysis is currently available only for mastodon.social, offering automated assessment of recent posts for potentially harmful content like hate speech or misinformation. (Covers D3.1 Requirement #6 (partially); D3.2 Functionality #6)
- *Exportable Visual Reports and Statistical Insights:* users can generate visual reports summarizing the moderation behaviour and content patterns of the analysed instance. These visualizations are limited to currently supported data but already support interpretation and communication of insights. (Will complete D3.1 Requirements #5 and #9; D3.2 Functionality #9)

3. Scientific Rationale

The design and implementation of the *Disinformation Detection in the Next-Generation Media* service (comprising of the two tools, namely Fediverse Network Moderation Tool and Mastodon Instance Inspector) described in this deliverable builds upon the assessment of state-of-the-art in next generation social media and generative AI as well as research results of WP2 as summarized in deliverables D2.1 *Report on technological and socio-behavioural characterization of next-generation social media* and D2.2 *Report on the assessment of the role of novel text and image generative AI methodologies in producing and combating disinformation*.

Deliverable D2.1 focused on next-generation social media, namely decentralised (federated) media, such as Mastodon or Bluesky and immersive virtual environments, such as Decentraland. The main outcomes impacting the design of the service were as follows:

- While immersive virtual worlds show promise for future social interaction, their current gaming-centric nature and predominantly young user base suggest it is premature to deploy comprehensive disinformation countermeasures in these spaces. As a result, **the Disinformation detection service focuses solely on federated social media**.
- Being decentralised, federated platforms like Mastodon present unique opportunities, but also challenges when addressing disinformation. On one hand, the structure of federated platforms supports effective **moderation through instance-level decisions about bans and labels**, which can be further analysed and utilized to get community assessment of various instances in the network. On the other hand, detection algorithms and policies can vary across instances, which hinders a coordinated response to disinformation - a service to analyse various instances in a unified way would thus be beneficial.
- Decentralised social media, such as Mastodon offer **a rich set of API endpoints for collecting public data** about instances, their content, and users, which is directly used in the design and implementation of the *Disinformation detection service*.

Deliverable D2.2 focused on text and image generative AI and its role in producing but also combatting disinformation. The main outcomes impacting the design of the service were as follows:

- Current state-of-the-art **LLMs (Large Language Models) can be used to generate disinformation that can be mistaken for human-written** especially in case of high-resource languages (with a slightly worse performance in low-resource languages).
- On the other hand, it is possible to **detect such machine-generated text when using multilingual fine-tuning and adversarial retraining** using various data augmentation techniques. We integrate such a machine-generated text detector into the *Disinformation detection service*.
- Other important approaches to help detect disinformation are **detection of check-worthy claims and retrieval of previously fact-checked claims**. These can also operate in multilingual settings and provide insights on the content spread in the network. Therefore, also these are integrated into the *Disinformation detection service*.
- **Fine-tuned smaller specialized models can still outperform general large models** used in zero-shot or even few-shot settings if there is enough data available for fine-tuning. This finding is important also considering the carbon footprint of the Disinformation detection service.
- Regarding the image generative AI, **its widespread accessibility has lowered the barrier for non-experts to generate synthetic images**, increasing the risk of its misuse to produce disinformation. While **it is possible to detect such synthetic images, it remains an unresolved challenge**, with no detection method emerging as clearly superior. Nevertheless,

the service (specifically Mastodon Instance Inspector) plans to connect with a synthetic image detection service MAAM, developed as a part of T4.3, as well as with other available synthetic image detection APIs integrated into MAAM to analyse the image content spread in the federated instances using a range of detection methods (this way mitigating the fact there is no single method to reliably detect everything).

The design of the service also reflects the main recommendations based on the assessment of the current state of next-generation social media and generative AI as summarized in the deliverable *D2.3 Adaptation of existing credibility and trust assessment and disinformation detection tools to next-generation social media and AI-generated content*. Specifically, it considers the following recommendations:

- *5.1 Recommendations for data access and processing:* The service uses available APIs accessing only publicly available data (5.1.1). It aims to minimise data that is fetched and processed (5.1.2) and it does not store any user data after the processing (5.1.3). The data are processed in a privacy-preserving way (5.1.4) and only aggregated outputs on the instance level are presented (5.1.5).
- *5.2 Recommendations for underlying classification models:* The content analysis modules included in the service were trained on robust evaluation datasets containing variable data from a range of relevant sources (5.2.1). They were also trained in a robust way (5.2.3) with their performance also already evaluated or planned to be evaluated on out-of-distribution data were possible (5.2.2). Since there is typically a trade-off between detection performance and inference costs (i.e., larger models typically have a better performance, but require more computational resources), we aim to select models that somehow optimize this trade-off, i.e., offer good enough performance at acceptable costs (5.2.4). All of the content analysis modules were already tested to some extent on data from federated social media (Mastodon) to detect drop in performance due to data shifts (5.2.5) with additional testing planned in future (as part of WP5).
- *5.4 Recommendations for user-facing AI-CODE services:* The service is designed to communicate results of analysis in understandable way providing means of setting classification thresholds where available (5.4.1). It also supports multi-modal content analysis (5.4.3) by integrating text analysis as well as image analysis/detection modules. Finally, it utilizes the existing fact-checked claims to detect disinformation narratives spread in the instances relying on the ClaimReview tagging system (5.4.4).

Based on the assessment of the current state of the art, **the proposed solution has the following main novelty and added value:**

- The service retrieves and analyses a **network of bans**, i.e., moderation decisions and reasonings to banning certain instances within the federated network. This functionality is particularly relevant in the context of the Fediverse, where moderation is handled independently by each instance. By aggregating and presenting these moderation signals, the service offers a useful perspective on how instances interact and respond to one another's content policies. This can support reputation assessment and robust moderation decisions across the decentralized network and provide valuable help in the presence of problematic instances.
- The service provides a range of **state-of-the-art network analysis and information propagation techniques** to analyse the instances in the Fediverse.
- The service uses a range of **state-of-the-art content analysis modules** developed (or

adapted) within the AI-CODE project including: machine-generated text detection, hate speech detection, check-worthy claim detection, or previously fact-checked claim retrieval.

- The service will interconnect with the MAAM service offering a **range of state-of-the-art methods for detection of synthetic and manipulated images**.

4. System Description

4.1 Prototype Architecture

The architecture defined for the AI CODE prototypes aims to offer a flexible and scalable foundation that supports both the development and deployment of the project's diverse set of services. To this end, a container-based infrastructure has been adopted, enabling modularization, reproducibility, and ease of integration.

The platform envisions two main deployment scenarios: a single-node setup for early-stage development and testing, and a multi-node setup for full-scale production environments. These scenarios are not merely conceptual, but have been practically validated through the actual containerization and deployment of services such as Mastodon_NetMod, providing a concrete implementation path that can be replicated by all partners.

In the single-node scenario, the architecture relies on Docker and Docker Compose to coordinate the deployment of various microservices within a single host. This approach is ideal for development and prototyping, as it minimizes infrastructure overhead while preserving isolation between services. Each service is containerized and configured via environment variables, with configuration files managed externally through .env files or passed directly into the container runtime.

A real example of this approach is the deployment of the Mastodon_NetMod service, which was pulled from a GitHub repository and built into a custom Docker image. Its execution was orchestrated alongside a MongoDB instance using a Docker Compose setup. Communication between services is managed through named networks, and traffic exposure is facilitated by tools such as Traefik or, for testing purposes, Nginx.

The architecture supports the inclusion of CI/CD pipelines even in this minimal setup, allowing automated image building and deployment through Jenkins or GitLab CI. This capability enhances the reproducibility of the setup and reduces manual effort for each iteration of the service.

In production scenarios, the architecture shifts towards a more robust, multi-node model, leveraging Kubernetes as the orchestration layer. Kubernetes enables automatic scaling, self-healing, and centralized management of deployments, making it ideal for ensuring high availability and long-term maintainability. Within this setup, services are distributed across multiple nodes, with each component—from load balancers to databases and monitoring agents—managed declaratively.

The architecture incorporates state-of-the-art technologies such as Cilium for advanced networking with eBPF, Rook Ceph and OpenEBS for flexible persistent storage, and Velero for disaster recovery. CI/CD is handled using GitOps approaches with ArgoCD and Helm, allowing continuous deployment

pipelines to be fully automated and version-controlled.

Observability is achieved using Prometheus and Grafana, while secret management and enhanced security can be integrated via Vault or Kubernetes-native solutions.

The transition from a single-node to a multi-node setup is designed to be smooth, as container definitions and service compositions remain compatible between Docker Compose and Kubernetes manifests.

To streamline integration for all project partners, a common baseline of pre-deployed tools has been proposed across both architectures. This includes databases such as PostgreSQL and MongoDB, cache systems like Redis, and messaging platforms such as Kafka or Redpanda.

These tools are provided out-of-the-box but remain fully optional and modular—partners may disable or replace components as needed to suit their own service requirements. This shared foundation significantly reduces onboarding time and ensures that all partners operate in compatible environments, simplifying testing, debugging, and collaborative development.

This architecture is not only technically sound but also practical, as shown by the deployment experience with Mastodon_NetMod and Mastodon Instance Inspector.

The deployment process followed best practices: using version-controlled Dockerfiles, setting up services via Docker Compose, exposing APIs through secure endpoints, and planning future steps such as CI/CD integration and improved monitoring with Prometheus and Grafana.

These lessons learned have been fed directly into the architectural definition, shaping a deployment strategy that is both replicable and extensible across the project's lifecycle.

To illustrate this dual-approach model, two diagrams representing the single-node and multi-node architecture will be provided in the final version of this deliverable.

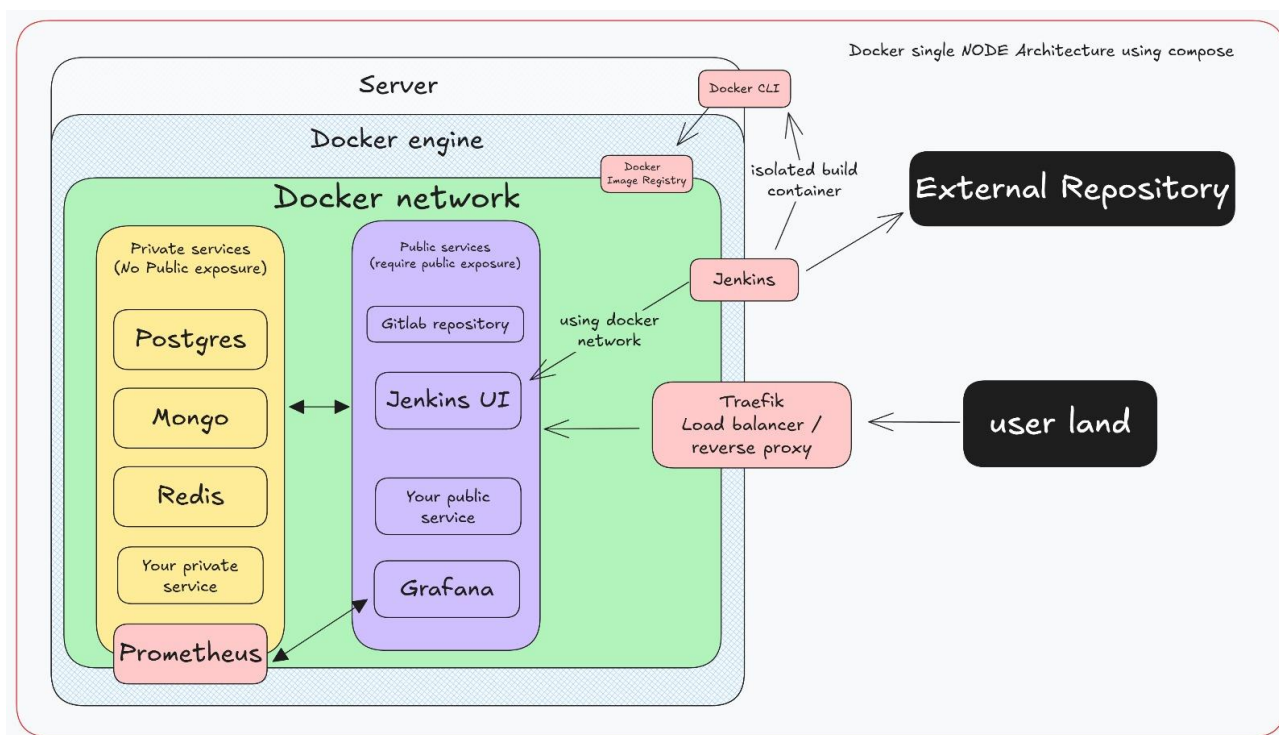


Figure 1 Prototype-Docker Single Node Architecture

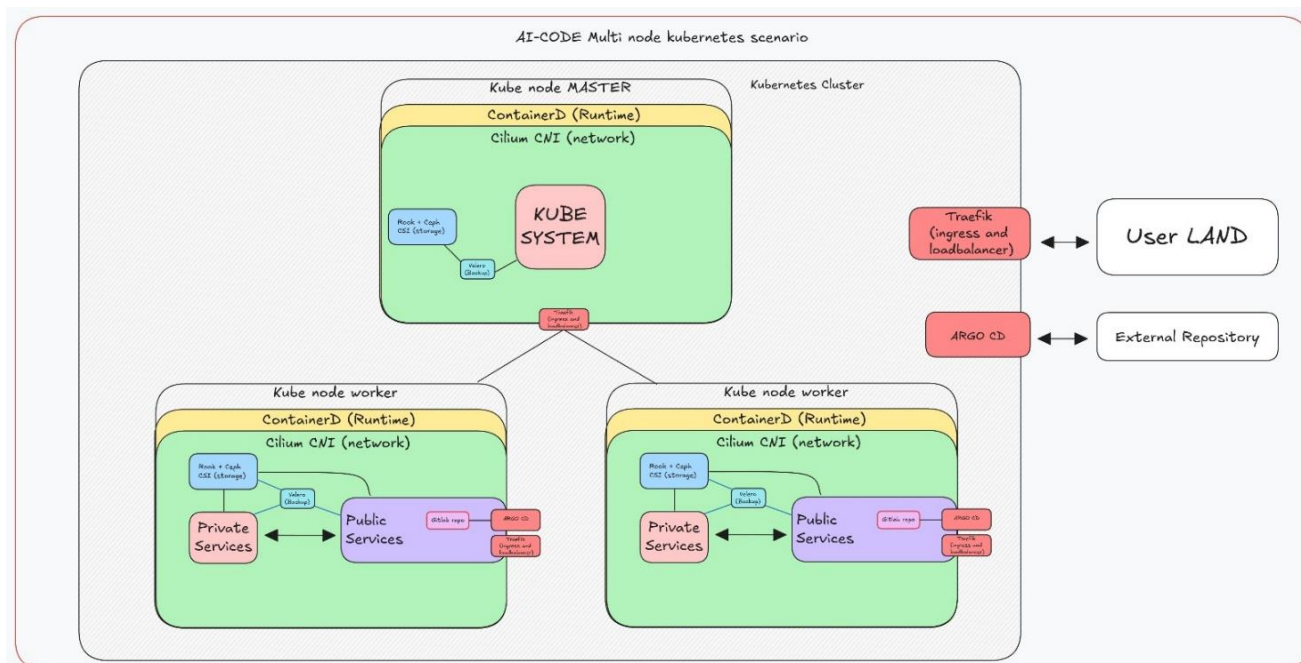


Figure 2 Prototype - Docker Multi-Node Architecture

4.2 Technical Description

As part of the AI-CODE project activities, DST deployed the services on a virtual machine running Ubuntu 22. The deployment was carried out through Docker containers, with service parameters passed via environment variables. The following steps describe the process in detail.

4.2.1 Usage and Access Information

To support integration and testing by project partners, a service endpoint has been made available through a secured reverse proxy accessible at:

<https://aicode.dstech.info>

The interface includes auto-generated documentation which allows users to trigger the exposed functionalities. By selecting an endpoint (e.g., a data collection or processing function) and using the “Try it out” option, users can initiate operations. Execution time may vary depending on the service complexity, with some processes requiring approximately **5 to 10 minutes** to complete.

4.2.1.1 Mastodon_NetMod Service

- Example of available endpoint with "Try it out" capability:

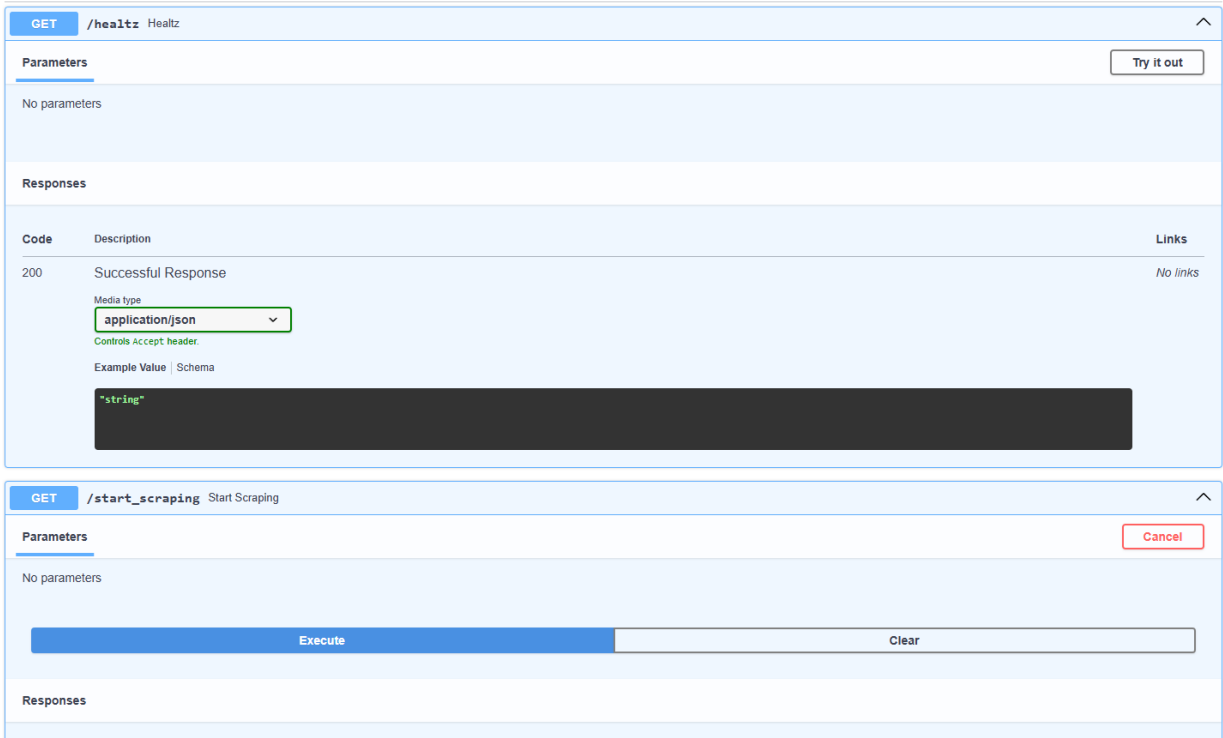


Figure 3 Example Endpoint NetMod

Sample result of an executed operation:

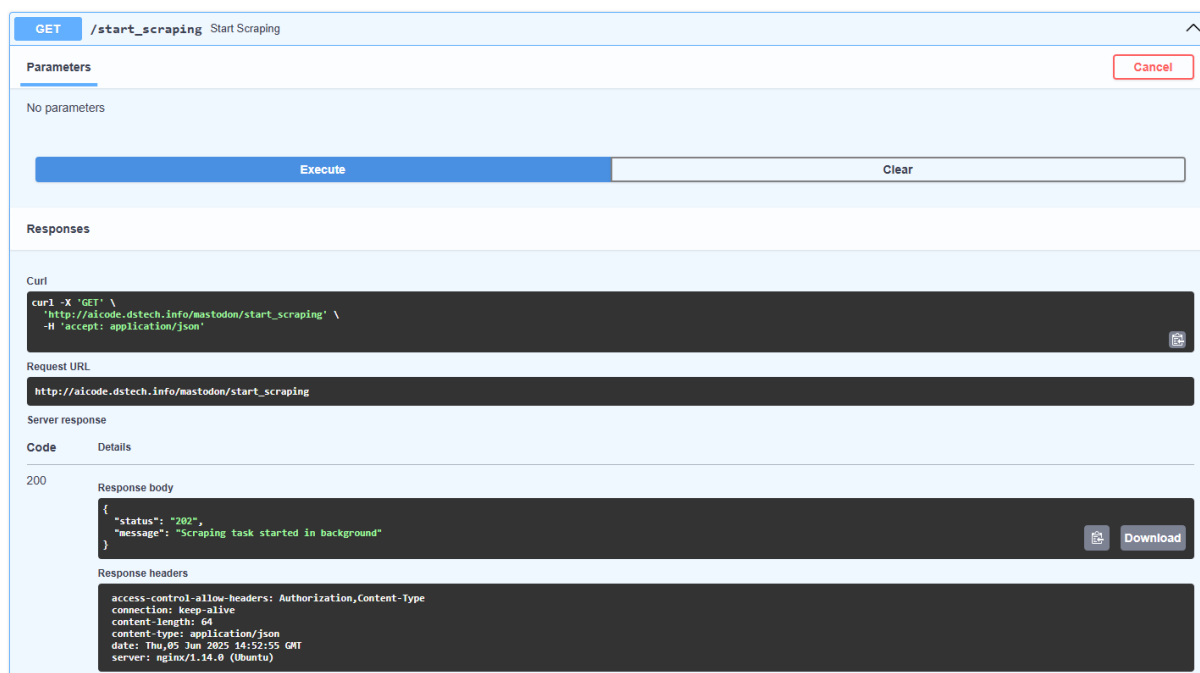


Figure 4 Sample executed Operation NetMod

4.2.1.2 Mastodon NetMod Use Case

The following use cases show how professionals might benefit from the current set of functionalities made available by the first prototype of the Fediverse Network Moderation Tool. The examples presented next demonstrate its potential usage in real-world scenarios, where collecting and accessing the moderation strategies enforced by Mastodon servers enables a better understanding of the Fediverse's dynamics. Further advancements for this prototype will enhance the user-tool interaction capabilities, as well as the amount of metadata can be extracted via this tool.

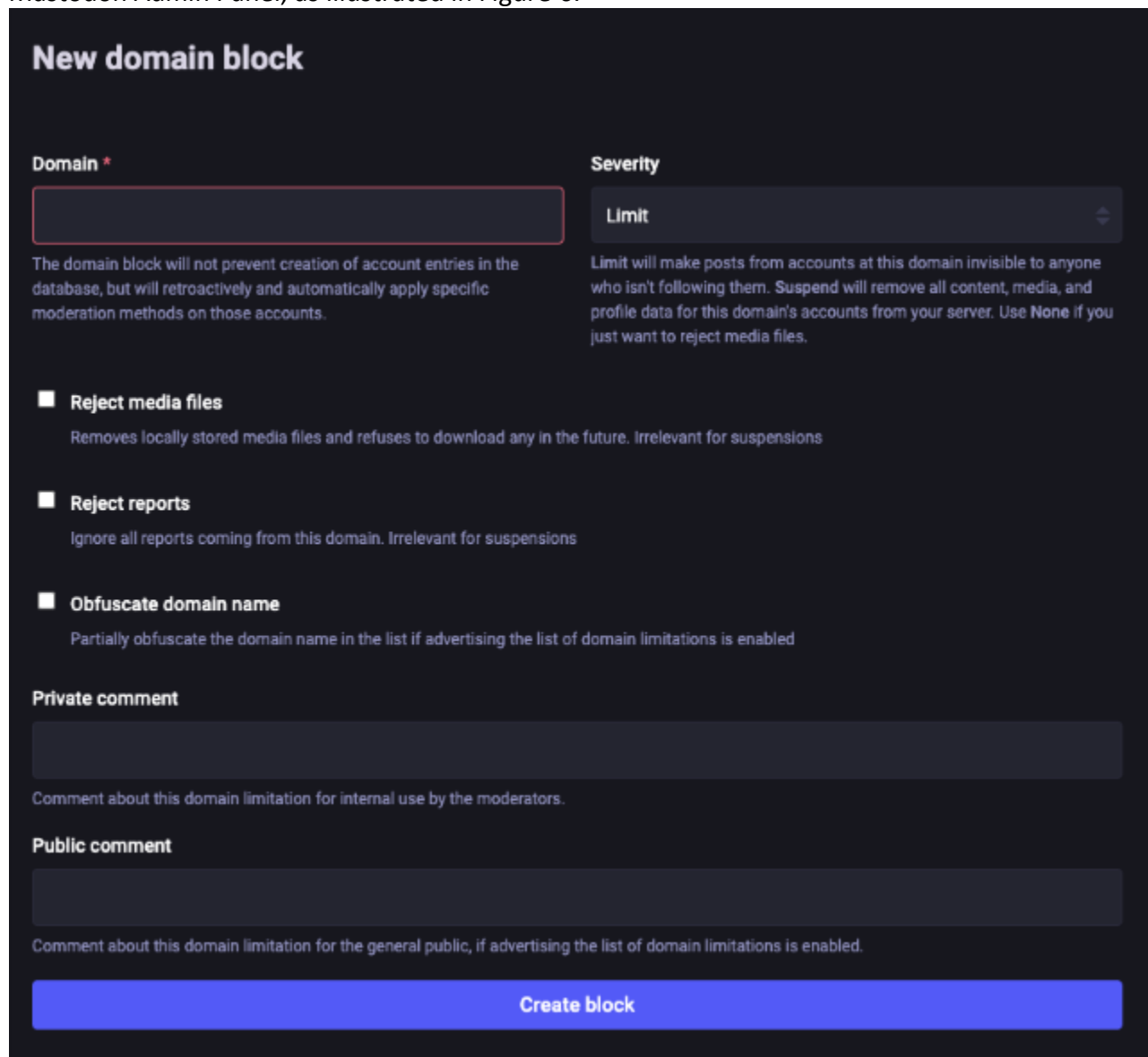
Use Case: Policy-makers Tracking Moderation Dynamics in the Fediverse

Sara, a policy-maker, is investigating moderation dynamics in the Fediverse to draft new policies about content spreading from potentially harmful instances. Using the Fediverse Network Moderation Tool, she is able to access the current landscape of moderation strategies enforced by online Mastodon instances. The tool allows her to identify instances that are frequently banned, along with the motivations for their moderation. Also, she can access temporal trends, gaining a deeper understanding of whether such strategies are consolidated through time or need further investigations due to anomalous peaks. Sara can hence take advantage from this tool to improve their understanding of the moderation dynamics in the Fediverse, and improve policy recommendations to enhance platform governance in Decentralized Social Media.

Use Case: Server Admins of a Media Company's Instance

John is the server admin for a media company's instance that recently joined the Fediverse. However, John is not aware of the potentially harmful instances living in this decentralized setting, and would like to keep the managed instance as safe as possible. To this aim, John can leverage the Fediverse Network Moderation Tool to discover which are the instances in the Fediverse that are typically banned, and also their motivation. Based on this information, he can hence decide to

prevent or limit interactions with these instances. For example, he can discover instances that are typically banned for fake news spreading, and manually add these to the enforced moderations of his instance to prevent harmful interactions. This can be easily done via the Mastodon Admin Panel, as illustrated in Figure 5. Similarly, he can decide to follow the entire moderation strategy enforced by a well-known instance (e.g., mastodon.social). To this aim, he can use the Fediverse Network Moderation Tool to fetch them, and export all rules as a single file, which can be imported via the Mastodon Admin Panel, as illustrated in Figure 6.



The screenshot shows the 'New domain block' form in the Mastodon Admin Panel. The form is dark-themed with white text. It has a title 'New domain block' at the top left. Below the title, there are two main sections: 'Domain' and 'Severity'. The 'Domain' section has a text input field with a red border and a red asterisk, indicating it is required. Below the input field, there is a description: 'The domain block will not prevent creation of account entries in the database, but will retroactively and automatically apply specific moderation methods on those accounts.' The 'Severity' section has a dropdown menu with 'Limit' selected. Below the dropdown, there is a description: 'Limit will make posts from accounts at this domain invisible to anyone who isn't following them. Suspend will remove all content, media, and profile data for this domain's accounts from your server. Use None if you just want to reject media files.' Below these sections, there are three checkboxes with labels and descriptions: 'Reject media files' (removes locally stored media files and refuses to download any in the future. Irrelevant for suspensions), 'Reject reports' (ignore all reports coming from this domain. Irrelevant for suspensions), and 'Obfuscate domain name' (partially obfuscate the domain name in the list if advertising the list of domain limitations is enabled). Below these checkboxes, there are two text input fields: 'Private comment' (comment about this domain limitation for internal use by the moderators) and 'Public comment' (comment about this domain limitation for the general public, if advertising the list of domain limitations is enabled). At the bottom of the form, there is a large blue button labeled 'Create block'.

New domain block

Domain *

Severity

Limit

The domain block will not prevent creation of account entries in the database, but will retroactively and automatically apply specific moderation methods on those accounts.

Limit will make posts from accounts at this domain invisible to anyone who isn't following them. Suspend will remove all content, media, and profile data for this domain's accounts from your server. Use None if you just want to reject media files.

☐ **Reject media files**
Removes locally stored media files and refuses to download any in the future. Irrelevant for suspensions

☐ **Reject reports**
Ignore all reports coming from this domain. Irrelevant for suspensions

☐ **Obfuscate domain name**
Partially obfuscate the domain name in the list if advertising the list of domain limitations is enabled

Private comment

Comment about this domain limitation for internal use by the moderators.

Public comment

Comment about this domain limitation for the general public, if advertising the list of domain limitations is enabled.

Create block

Figure 5 Mastodon Admin Panel for moderating a single instance

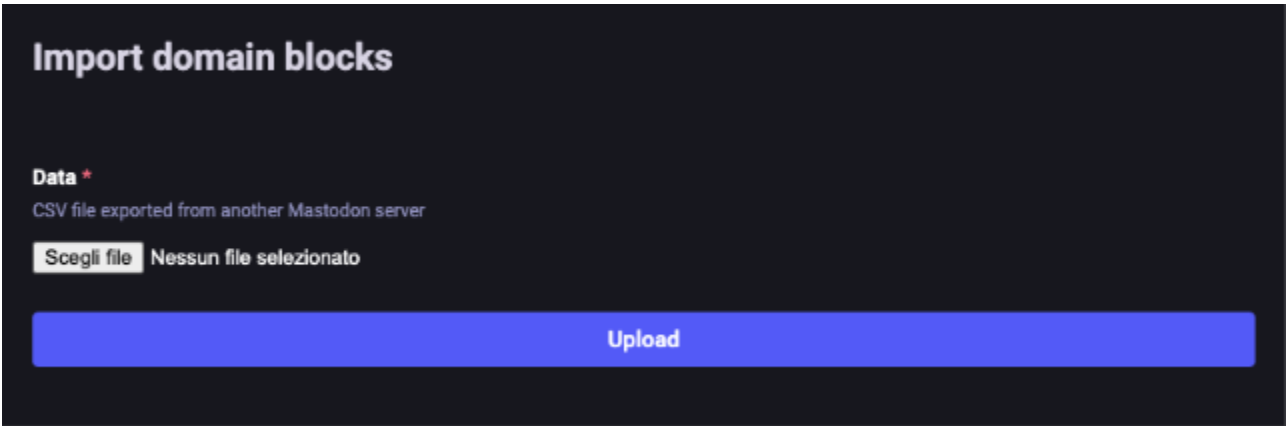


Figure 6 Mastodon Admin Panel for importing moderation strategies from other instances

4.2.1.3 Mastodon Instance Inspector Service

- Example of available endpoint (/download_output) with documentation:

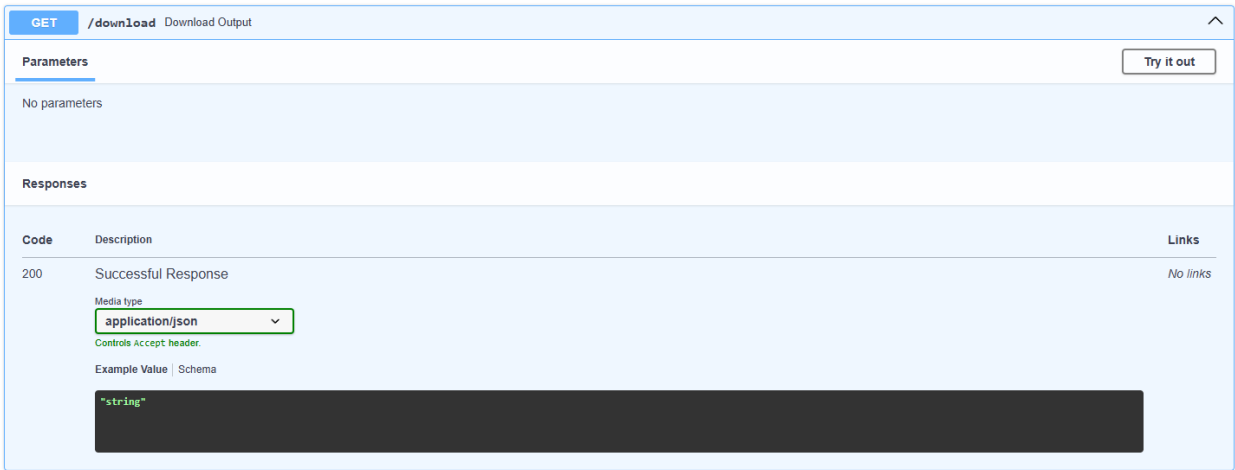


Figure 7 Example endpoint Instance Inspector

- Example of a generated graphical output:

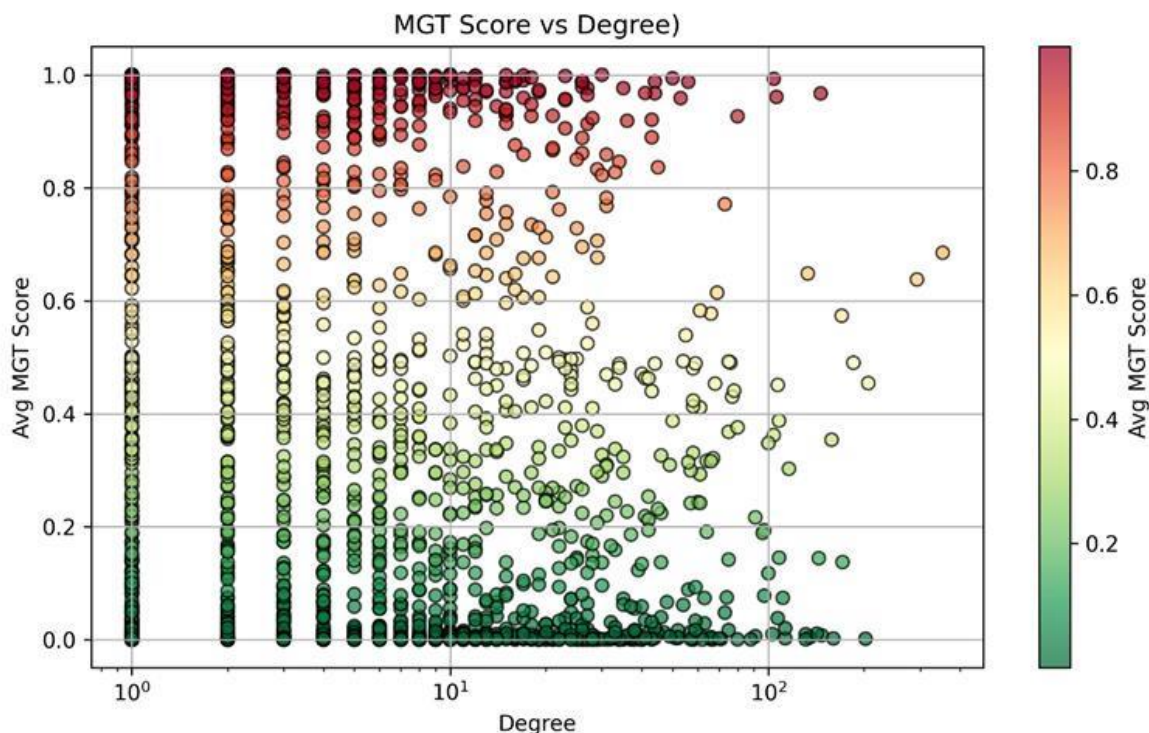


Figure 8 MTG Score vs Degree Example graphical

4.2.1.4 Mastodon Instance Inspector Use Cases

The following usage scenarios illustrate how professionals might benefit from the Mastodon Instances Inspector's current capabilities. While the prototype is presently limited to content analysis for a single instance (mastodon.social), these examples demonstrate the potential value of its analytical features in real-world tasks such as credibility assessment, content monitoring, and instance evaluation. Future releases will expand support to multiple instances and additional content types, further enhancing these workflows.

Use Case: Investigative Journalist Exploring a High-Activity Instance

Emily, an investigative journalist, is researching online reactions to recent political developments. Using the Mastodon Instances Inspector, she explores mastodon.social, one of the largest and most active instances on the Fediverse.

She begins by examining content-level signals:

- ⊄ A noticeable number of posts are flagged as **very likely machine-generated**, suggesting automated amplification activity

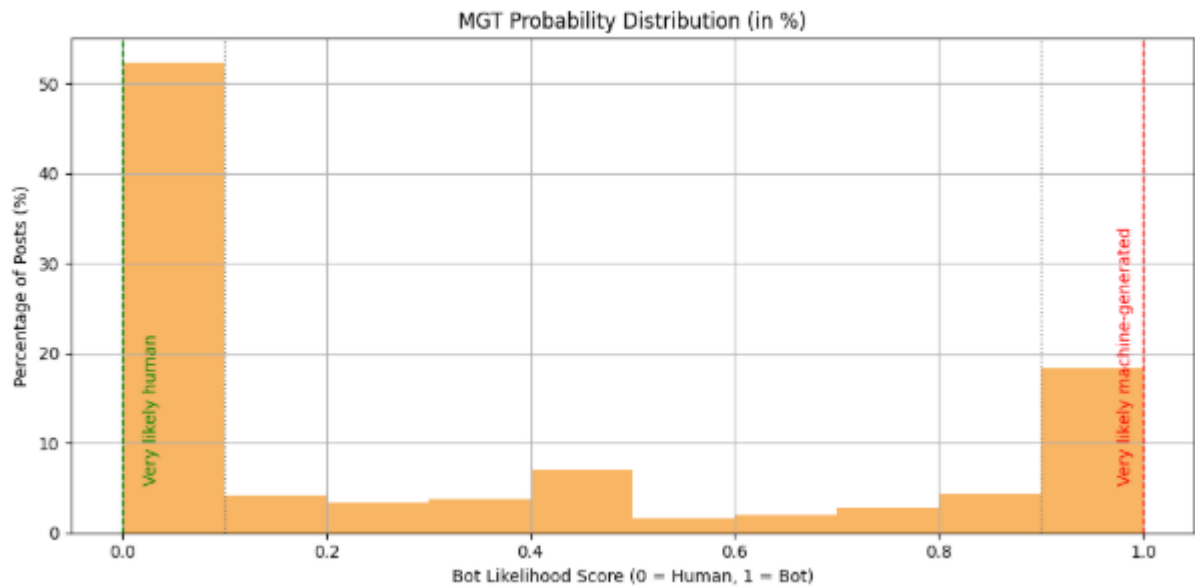


Figure 9 MGT Probability Distribution Graph

⊄ A small but clear segment of posts is scored as **check-worthy**, raising interest for further investigation by her team.

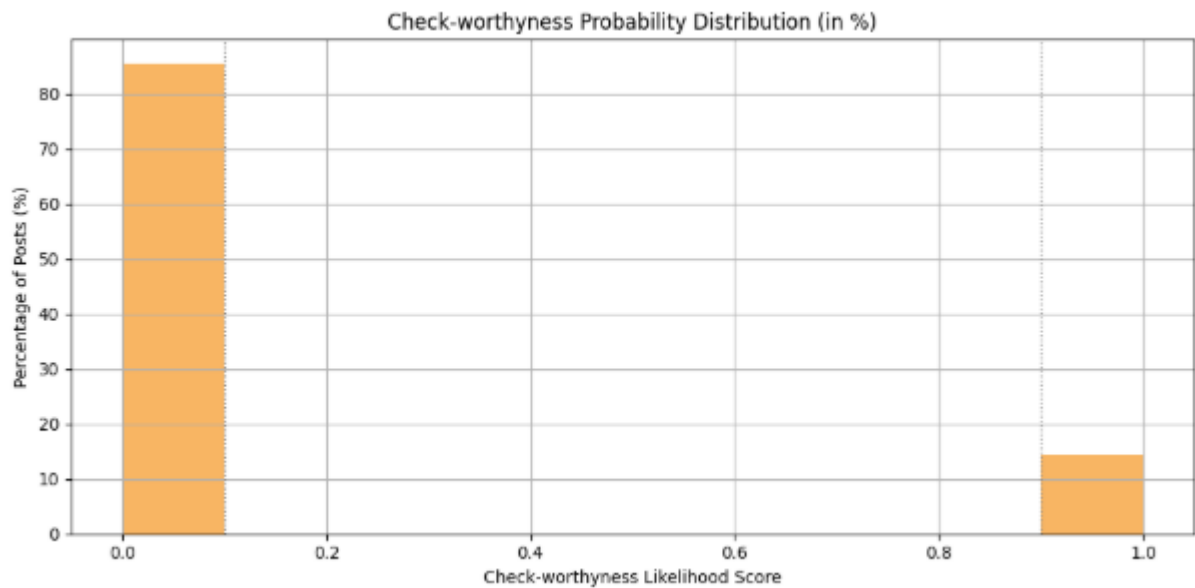


Figure 10 Check-worthiness Probability Distribution

⊄ **Top hashtags** include political themes (#trump, #ukraine), helping her identify trending narratives.

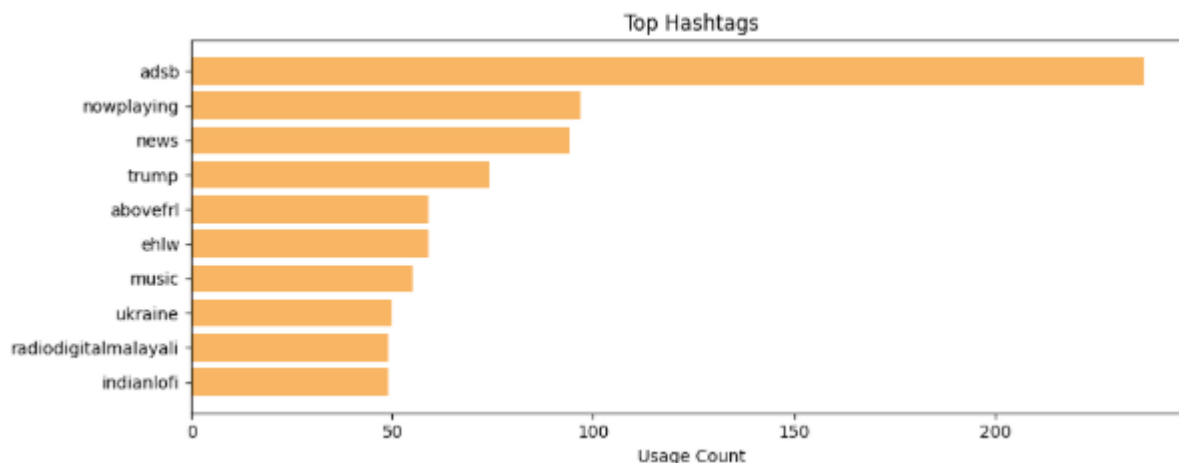


Figure 11 Top Hashtags

Emily also reviews moderation data for mastodon.social in the instance network graph. Although the current prototype only supports one instance, the visual summary gives her an idea of its position in the broader moderation landscape. This enables her to confidently describe the nature of discourse on mastodon.social, and use the visual reports to inform her story's framing.

Use Case: Communication Manager Assessing Instance Suitability for Brand Engagement

User: Julia, Digital Marketing Lead

Goal: Evaluate the suitability of a Mastodon instance for safe and values-aligned brand presence.

Context:

Julia's team is exploring the Fediverse as a space for launching a brand awareness campaign tied to ethical fashion. While they can't yet analyze multiple instances, the **Mastodon Instances Inspector** allows Julia to explore the largest instance, mastodon.social, as a representative example of user dynamics and platform tone.

What Julia finds using the Inspector:

- ✎ **Language Distribution:** The instance is predominantly English-speaking, but includes significant multilingual activity, supporting international reach.
- ✎ **Trending Hashtags:** Recent topics include social issues, tech, and activism — closely aligned with her brand's values (see hashtag bar chart).
- ✎ **Content Safety Metrics:** Only a small portion of posts are tagged NSFW, suggesting a low risk of reputational harm in typical user interactions (see NSFW ratio chart).

Outcome:

While Julia cannot yet compare multiple instances, the insights from mastodon.social give her a reliable benchmark. The data visualizations support her team's internal discussions about tone, content risk, and campaign suitability. When multi-instance support is added in future versions, her team plans to extend this approach to find smaller, niche communities aligned with the brand.

4.3 Ethical Design Considerations

The design of the Disinformation Detection service in Next-Generation Media (T4.1) has integrated ethical considerations from the early development stages, following the ethical requirements specified in D3.1 (Section 10.5). These requirements were interpreted through the lens of technical feasibility and ethical impact, particularly in the context of decentralized environments and AI-assisted content inspection.

The table below summarizes the ethical requirements addressed by the current prototype, including examples of their implementation. Where full implementation was not feasible at this stage, we outline planned or partial measures.

#	Ethical requirements	How is the requirement addressed in the current prototype design?
HAO-01	It shall be apparent to the user interacting with the application that they are interacting with an AI system.	Our service will provide aggregated results, and the user will not directly interact with the AI system.
HAO-02	The capabilities of the application shall be communicated clearly to the user.	Detailed documentation will outline what the system can and cannot do, including any limitations or assumptions in its analysis.
PDG-01	The application shall minimize the amount of data collected on users.	The system will not collect persistent user data. On-demand content analysis is triggered explicitly by the user and does not store user-identifying information.
TSA-03	The data used to build the application shall be up-to-date, of high quality, complete and representative of the deployment environment.	For on-demand analysis, the system fetches content in real time from publicly available posts. Data is processed immediately and discarded after use to ensure freshness, privacy, and relevance. (APIs are to be implemented)
FAI-02	The application shall take the diversity and representativeness of the end user into account.	Interface design and content categories will reflect a range of languages, cultures, and moderation contexts. User testing with diverse stakeholder groups will inform design iterations to ensure inclusivity and accessibility.

Table 3 Ethical considerations.

Following the recommendations of the Ethics Advisory Board, the following measures will be accounted on the development of the service.

- Misclassification Risks: users will be informed through disclaimers that moderation data is crowd-sourced and may change over time.
- User identification: no usernames will be shown.

5. Next Steps

The study of the state of the art in WP2 and the analysis of the requirements in WP3 (in particular in D3.1 with the co-creation workshops and with the technical requirements of D3.2) have allowed

us to obtain a very accurate and functional prototype of the services in basic operations. In particular, the technical requirements obtained from WP3 have allowed us to size, in principle, the necessary resources and the best technologies to be used in the development of the prototypes at present, but also to lay a good foundation for the development of the complete services that will start with WP5. In addition, the application architecture, both single-node and multi-node (if necessary) will allow us to have a lot of elasticity and modularity both in the prototypes and in the final versions. The micro-services management, via docker, will allow us to use the services in stand-alone mode or in parallel (even intertwining heterogeneous functions via API).

The next step, namely the validation and complete implementation of the services, hence WP5, will start from the architecture just described, which will then be the basis for the integration of all the tools. Having developed everything separately, with the use of APIs, it will be very easy to move from a simpler configuration to a complex multi-node one.

Based on this technical foundation, the next development steps will focus on expanding the Mastodon Instance Inspector's features to better meet user needs and tackle key challenges from earlier work packages. In particular, next upgrades will expand its analytical capabilities, improve data coverage, and introduce new mechanisms to monitor information flow and disinformation risk across the Fediverse.

1. **Expansion of Content Analysis Capabilities:** the current AI-assisted content analysis module, limited to textual posts from mastodon.social, will be extended to support a broader set of instances and media types, including images. This will support more effective monitoring of misinformation signals across the Fediverse.
2. **Scalable Moderation Data Aggregation:** the service will automate the collection of content moderation metadata across multiple instances. This enhancement will facilitate ongoing, real-time updates and allow users to query moderation patterns more comprehensively.
3. **Information Flow and Infodemic Risk Tracking:** a major upcoming feature is the implementation of network-wide tracking of information diffusion. This will allow users to trace how domains and URLs propagate across instances and to assess the credibility and clustering of sources. New metrics such as an Infodemic Risk Index will be introduced to evaluate the vulnerability of specific instances to coordinated disinformation campaigns. Visualizations of source networks and spread dynamics will help journalists and researchers better understand the flow and origin of misleading content.
4. **User Experience and Report Generation:** future iterations will further refine the user interface, including support for customizable filters (e.g., by moderation behavior, bot activity, or topic) and dynamic network exploration tools. Additionally, the report generation feature will be extended to include cross-instance comparisons, making it easier for journalists, fact-checkers, and communication professionals to summarize and communicate findings in a clear, evidence-based manner.
5. **Enhanced Categorization and Analytics for Moderation Tool:** next versions will focus on improving the categorization of moderation dynamics, and will consider providing users with some aggregated statistics to let end-users see moderation dynamics for a current period (e.g., day/month) at a glance before interacting with the service.
6. **Historical moderation trend analytics:** historical information on moderation dynamics: the service will provide users with historical data, including the number of times servers have been flagged for specific reasons, and the most-frequently banned servers for each reason.

6. Conclusions

This deliverable presented the first functional prototypes of two core services developed under Task 4.1 of the AI-CODE project: the Fediverse Network Moderation Tool and the Mastodon Instance Inspector. These tools address critical challenges posed by decentralized social networks, particularly the fragmented moderation landscape and the growing threat of disinformation in federated platforms like Mastodon.

By combining moderation dynamics analysis with AI-assisted content inspection, the services offer a unified infrastructure to enhance transparency, trustworthiness, and informed decision-making across the Fediverse. The Fediverse Network Moderation Tool enables the collection and querying of moderation data to support community safety, while the Mastodon Instance Inspector provides analytical capabilities for evaluating the credibility and behavioural patterns of individual instances. Together, they form a complementary suite designed to support journalists, fact-checkers, researchers, policymakers, and community moderators.

The work carried out so far builds on a strong technical foundation informed by stakeholder requirements (D3.1 and D3.2) and scientific insights from WP2. The adoption of a microservices architecture, accessible via Docker and proxy endpoints, ensures modularity, scalability, and integration readiness for future developments.

While the prototypes are already operational and publicly accessible, they are also purposefully designed for evolution. The roadmap outlined for WP5 includes expansion to additional instances and content types, integration with image disinformation detection (via MAAM), and enhanced user interfaces with improved visualization and reporting features. Moreover, the tools will incorporate advanced functionalities such as information flow tracking, historical trend analysis, and real-time moderation dynamics monitoring—further solidifying their role as essential instruments in combatting online disinformation in decentralized environments.

In conclusion, this first development phase marks a significant milestone in AI-CODE's ambition to foster continuous trust in digital ecosystems. The results achieved provide not only a tangible demonstration of technical feasibility but also a clear direction for upcoming implementation and validation efforts. The services developed in T4.1 lay the groundwork for robust, trustworthy, and scalable AI-driven solutions that can support the responsible evolution of next-generation social media platforms.

7. Annex: Deployment and Configuration Details for AI-CODE Services

This appendix provides a comprehensive guide for setting up and deploying the two primary services within the AI-CODE prototype: **Mastodon_NetMod** and **Mastodon Instance Inspector**.

7.1 A.1 Initial Configuration

Deployment was performed on an Ubuntu 22.04 environment with Docker and Docker Compose installed:

```
sudo apt update && sudo apt upgrade  
  
sudo apt install docker docker-compose git  
  
sudo service docker start  
  
sudo usermod -aG docker $USER
```

7.2 A.2 Mastodon_NetMod Service

7.2.1 A.2.1 Dockerfile

7.3 Stage 1: Build

```
FROM python:3.9-slim AS builder  
WORKDIR /app  
COPY requirements.txt .  
RUN pip install --no-cache-dir --prefix=/install -r requirements.txt  
COPY . .
```

7.4 Stage 2: Runtime

```
FROM python:3.9-slim  
WORKDIR /app  
COPY --from=builder /install /usr/local  
COPY . .  
CMD ["fastapi", "dev", "./main.py", "--port", "8081", "--host", "0.0.0.0"]
```

7.4.1 A.2.2 Docker Compose Configuration

```
version: '3.8'  
networks:  
  traefik:  
    name: traefik_default  
    external: true  
  
services:  
  app:
```

```
image: mastodon:latest
volumes:
  - ./app
environment:
  - config.json=/app/config.json
depends_on:
  - aicode_mongodb
networks:
  - traefik
labels:
  - 'traefik.enable=true'
  - 'traefik.http.routers.app.rule=PathPrefix(/mastodon)'
  - 'traefik.http.routers.app.entrypoints=web'
  - 'traefik.http.services.app.loadbalancer.server.port=8081'
```

```
aicode_mongodb:
  image: mongo:4.4.6
  container_name: mongodb
  ports:
    - "27017:27017"
  volumes:
    - mongo_data:/data/db
  networks:
    - traefik
```

```
volumes:
  mongo_data:
```

7.5 A.3 Mastodon Instance Inspector Service

7.5.1 A.3.1 Dockerfile

7.6 Stage 1: Build

```
FROM python:3.10 AS builder
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir --prefix=/install -r requirements.txt
COPY . .
```

7.7 Stage 2: Runtime

```
FROM python:3.10
WORKDIR /app
```

```
COPY --from=builder /install /usr/local
COPY . .
CMD ["fastapi", "dev", "./main.py", "--port", "8081", "--host", "0.0.0.0"]
```

7.7.1 A.3.2 Docker Compose Configuration

```
version: '3.8'
networks:
  traefik:
    name: traefik_default
    external: true

services:
  prototype:
    image: prototype:latest
    volumes:
      - ./app
    networks:
      - traefik
    labels:
      - 'traefik.enable=true'
      - 'traefik.http.routers.prototype.rule=PathPrefix(/prototype)'
      - 'traefik.http.routers.prototype.entrypoints=web'
      - 'traefik.http.services.prototype.loadbalancer.server.port=8081'
```

7.8 A.4 Service Exposure via Traefik

Both services are exposed through the shared external network `traefik_default`. Routing is handled by Traefik using labels defined in the Docker Compose files.

Endpoints:

- Mastodon_NetMod → <https://aicode.dstech.info/mastodon>
- Mastodon Instance Inspector → <https://aicode.dstech.info/prototype>

Each service listens on internal port 8081 and is routed externally via Traefik path prefixes.

7.9 A.5 Service Startup

To launch the services:

```
docker-compose up -d
```

Once running:

- Mastodon_NetMod exposes:
https://aicode.dstech.info/mastodon/start_scrape?instance=https://mastodon.uno
- Mastodon Instance Inspector provides OpenAPI docs at:
https://aicode.dstech.info/prototype/docs#/default/download_output_download_get

7.10 A.6 Planned Enhancements

- Integration with CI/CD pipelines (e.g., GitLab CI or Jenkins)
- Hosting images on a private Docker registry
- MongoDB access restricted to localhost with SSH tunneling
- Application of firewall rules and access control policies
- Observability setup with Prometheus and Grafana for monitoring

7.10.1.1 Backend Database Access (MongoDB)

The backend service uses MongoDB, which is deployed securely with access limited to localhost:27017. External access is permitted via an SSH tunnel, with traffic routed through port 13022 on the public network, and redirected internally to port 22 of the virtual machine hosting MongoDB.

To create a tunnel, the following command should be used:

```
ssh -p 13022 -L 27017:localhost:27017 <username>@aicode.dstech.info
```

A non-sudo user account (aicode-partners) has been configured specifically for partners. Credentials are available via the Passbolt password manager and can be accessed after connecting to the company VPN.

Once connected via SSH, MongoDB can be accessed locally on the client system with:

```
mongosh mongodb://localhost:27017
```

This setup enables encrypted, controlled access to the database, without exposing it directly to the internet.

7.10.1.2 Notes on Proxy and Service Availability

A reverse proxy is active and configured to allow HTTPS traffic to the hosted services. In some cases, timeouts may occur depending on the nature of the request or service load. Partners are encouraged to report such occurrences and provide details about their intended usage so that appropriate tuning or support can be provided