Funded by the European Union
Horizon Europe
(HORIZON-CL4-2021-HUMAN-01-27
AI to fight disinformation)

Ref. Ares(2024)6917135 - 30/09/2024

www.ai4trust.eu

AI4TRUST

D5.5

AI4TRUST Platform v1

PARTNERS

FONDAZIONE BRUNO KESSLER · CERTH CENTRE FOR RESEARCH & TECHNOLOGY HELLAS · UNIVERSITÀ DI TRENTO · NATIONAL CENTRE FOR SCIENTIFIC RESEARCH "DEMOKRITOS" · CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE · POLITECNICA · GDI Global Disinformation Index · sky

SAHER EUROPE · DEMAGOG · MALDITA.ES · ASTIKI MI KERDOSKOPIKI ETAIRIA KENTRO KATAPOLEMISIS TIS PARAPLIROFORISIS / CIVIL NON-PROFIT COMPANY KENTRO KATAPOLEMISIS TIS PARAPLIROFORISIS · EURACTIV · ASOCIATIA DIGITAL BRIDGE · EUROPEJSKIE MEDIA SP ZOO · FINCONS GROUP · UNIVERSITY OF CAMBRIDGE

Project acronym	AI4TRUST
Project full title:	AI-based-technologies for trustworthy solutions against disinformation
Grant info:	ID 101070190-AI4TRUST
Funding:	EU-funded under Digital, Industry, and space Overall budget € 5.950.682,50
Version:	1.0
Status	Final version
Dissemination level:	Public
Due date of deliverable:	30/09/2024
Actual submission date:	30/09/2024
Work Package:	WP5
Lead partner for this deliverable:	FINC
Partner(s) contributing:	FBK
Main author(s):	Marco Giovanelli (FINC), Gabriel H. Carraretto (FINC)
Contributor(s):	Marcello Paolo Scipioni (FINC), Matteo Saloni (FBK)
Reviewer(s):	Riccardo Gallotti (FBK), Raman Kazhamiakin (FBK), Serena Bressan (FBK), Danilo Giampiccolo (FBK)

Summary of modifications

VERSION	DATE	AUTHOR(S)	SUMMARY OF MAIN CHANGES
1	08/07/2024	Marco Giovanelli (FINC), Gabriel H. Carraretto (FINC)	First draft
2	12/09/2024	Marco Giovanelli (FINC), Gabriel H. Carraretto (FINC), Matteo Saloni (FBK)	Add contributions from technical partners and consolidate the deliverable
3	18/09/2024	Serena Bressan (FBK)	Review of the first semi-final version of the deliverable
4	27/09/2024	Marco Giovanelli (FINC), Gabriel H. Carraretto (FINC)	Finalise the deliverable according to the internal review
5	30/09/2024	Riccardo Gallotti (FBK), Raman Kazhamiakin (FBK), Serena Bressan (FBK), Danilo Giampiccolo (FBK)	Final review of the deliverable text by the coordinator before submission

Table of contents

List of acronyms

ACRONYMS	MEANING
AI	Artificial Intelligence
API	Application Programming Interface
CIB	Coordinated Inauthentic Behaviour
DAG	Directed Acyclic Graph
DoA	Description of Action
DWS	Disinformation Warning System
IP	Intellectual Property
PS	Platform Specification
SW	Software
UI	User Interface
WP	Work Package

List of figures

List of tables

Executive summary

This document represents **Deliverable D5.5, titled "AI4TRUST Platform v1"** for the Horizon Europe project AI4TRUST "AI-based technologies for trustworthy solutions against disinformation". D5.5 is the second deliverable of **Work Package 5 (WP5) - "Technical implementation of the platform & Security Framework"**. It is closely interconnected with other Work packages, specifically **WP2** ("Methodological design, data gathering and pre-processing"), **WP3** ("AI-driven data analysis methods"), **WP4** ("Human-Centred Explainability, Interpretation and Policy"), and **WP6** ("Piloting, Assessment & Fact-checking").

The deliverable marks the **initial release of the AI4TRUST platform**. It also formalises the availability of the **main technical components developed in WP3**. This first version of the platform will be tested by consortium end users (i.e., fact-checkers and journalists) during the **first piloting phase in WP6**, starting in October 2024.

The deliverable is structured as follows:

- **Introduction:** an overview of the platform and its objectives;
- **Platform Overview:** a detailed description of the platform's architecture and functionality;
- **Front-end:** this section covers the user-facing components, such as login, file upload, and the analysis modules (i.e., image, video, text, and audio processing);
- **Back-end:** the technical implementation, including the OpenAPI specifications for platform operations, authentication, and media analysis (i.e., image, video, text, and audio);
- **Infrastructure:** it focuses on the system structure, repository organisation, versioning strategies, branching, documentation templates, and issue tracking mechanisms;
- **Integration and Deployment:** it describes the steps for the local integration and the deployment of the platform, ensuring smooth operation across different environments;
- **Conclusions and Recommendations:** the deliverable closes with a summary of key findings and future recommendations for improving the platform.

This initial version of the AI4TRUST platform lays the groundwork for the project goal of **combating disinformation through AI-driven solutions**, while ensuring compliance with privacy and ethical standards. The **insights gained from the upcoming piloting phase** will further **refine the platform capabilities**.

1. Introduction

The **AI4TRUST** "AI-based Technologies for Trustworthy Solutions Against Disinformation" project is a pivotal initiative aimed at **bolstering the human capacity to counter misinformation and disinformation** across the European Union (EU). This ambitious project seeks to **empower researchers, fact-checkers, media practitioners, and policy makers** by equipping them with **cutting-edge AI technologies** (WP3). These technologies are designed to address the multifaceted challenge of mis/disinformation through a comprehensive approach that encompasses **multichannel, multilingual, and multimodal monitoring**. Specifically, AI4TRUST targets the detection and recording of false information across diverse online social media platforms and traditional news outlets, ensuring coverage of about 70% of EU media and user interactions.

The project's core objectives are threefold: 1) to enable **robust monitoring** across various communication channels, languages, and content types; 2) to **assess the risks** associated with unreliable information consumption; and 3) to **foster a trustworthy online environment**. This environment will facilitate **collaboration** among researchers, fact-checkers, media practitioners, and policymakers, aiding in the creation and dissemination of **accurate information and effective counter-narratives**, while simultaneously labelling and addressing misinformation and disinformation.

Deliverable D5.5, titled **"AI4TRUST Platform v1"**, marks a significant milestone in this endeavour. As the second deliverable of Work Package 5 (WP5) - "Technical Implementation of the Platform & Security Framework", D5.5 represents the **initial release of the AI4TRUST platform**. This deliverable encompasses a detailed presentation of the platform functionality, which is crucial for understanding its capabilities and operational framework.

The document is structured to provide a comprehensive overview of the platform. It begins with an introduction to the platform and its objectives, followed by an exploration of the current version of the platform architecture in the **"Platform Overview" section**. The **"Front-end" section** details the user-facing components, including login processes, file upload mechanisms, and analysis modules for image, video, text, and audio content. The **"Back-end" section** focuses on the technical implementation, including OpenAPI specifications for various platform operations. Further, the **"Infrastructure" section** outlines the system organisational structure, repository management, versioning, and issue tracking. Finally, the **"Integration and Deployment" section** describes the processes for local integration and deployment to ensure seamless operation across different environments, while the **"Conclusions and Recommendations" section** summarises key findings and suggests future improvements.

This first version of the AI4TRUST platform lays the foundation for the project overarching goal of mitigating mis/disinformation through AI-driven solutions. The **forthcoming piloting phase** (WP6) will provide valuable insights that will refine the platform capabilities.

2. Platform Overview

The first version of the AI4TRUST platform focuses on the "AI-driven data analysis methods". These methods provide the user with textual, audio and visual analysis tools that can check in detail single news items (see Figure 1). In practice, users can provide to the AI4TRUST platform either a text - e.g., the textual content from a "Facebook" post - or a link to an image/audio/video - e.g., the link of a "YouTube" video - in order to obtain the results of the analyses carried out.

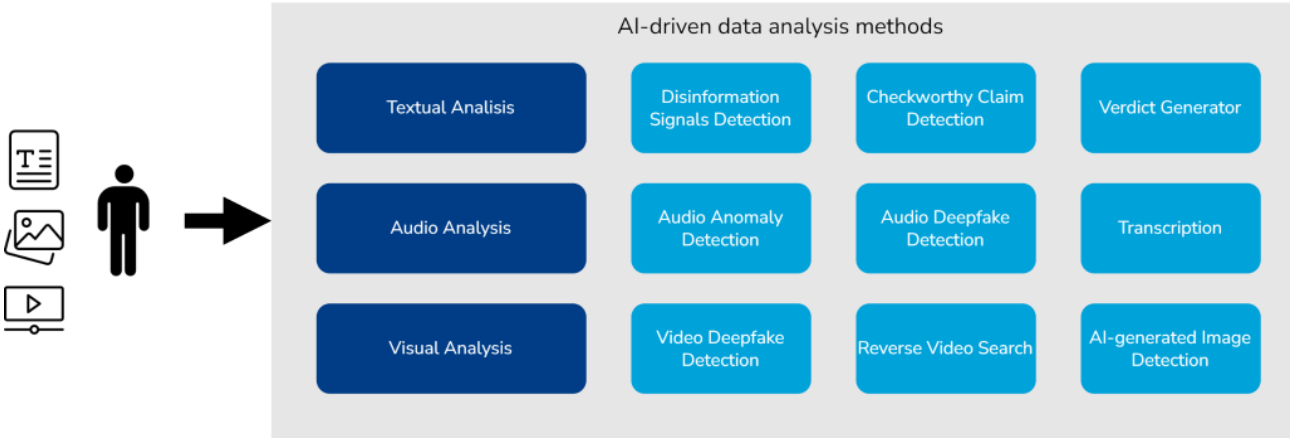

Figure 1 - Textual, audio and visual analysis tools of the first version of the AI4TRUST platform

In detail, the tools offered to the user are:

- **Disinformation Signals Detection**: This tool analyses text content, detecting instances of hate speech, offensive language, and clickbait (e.g., use of sensational, exaggerated, or ambiguous language, Over-the-Top/"Catchy" Headlines).
- **Check-worthy Claim Detection**: This tool indicates whether the text is worthy of verification and assigns a corresponding score.
- **Verdict Generation**: This tool helps users verify the accuracy of a given claim by analysing a reliable information source (such as an article or report) related to the claim's topic by providing a verdict indicating whether the claim is true or false, along with the reasoning behind this conclusion.
- **Audio Anomaly Detection**: This tool analyses the audio segment by segment and identifies anomalous segments (segments which might comprise splicing points or which might have been generated using AI).
- **Audio Deepfake Detection**: This tool analyses the audio as a whole and indicates whether the audio is real or was fully generated using AI.
- **Audio Transcription**: This tool transcribes into text the spoken content in the input audio or video file.
- **Video Deepfake Detection**: This tool analyses a video through a set of AI detectors and provides a probability of the video being a deepfake.

- **Reverse Video Search**: This tool checks whether near-duplicates of a video are present on the Web, and debunks fakes that are based on video re-use in a different context.
- **AI-generated Image Detection**: This tool analyses a video through a set of AI detectors and provides a probability of the video being a deepfake.

As shown in Figure 2, to enable these tools, the first version of the AI4TRUST platform envisages a frontend (that offers the functionalities to the user), a back-end (that is responsible to collect the frontend requests and communicate properly with the corresponding service), and a number of AI4TRUST services (performing the actual analysis).

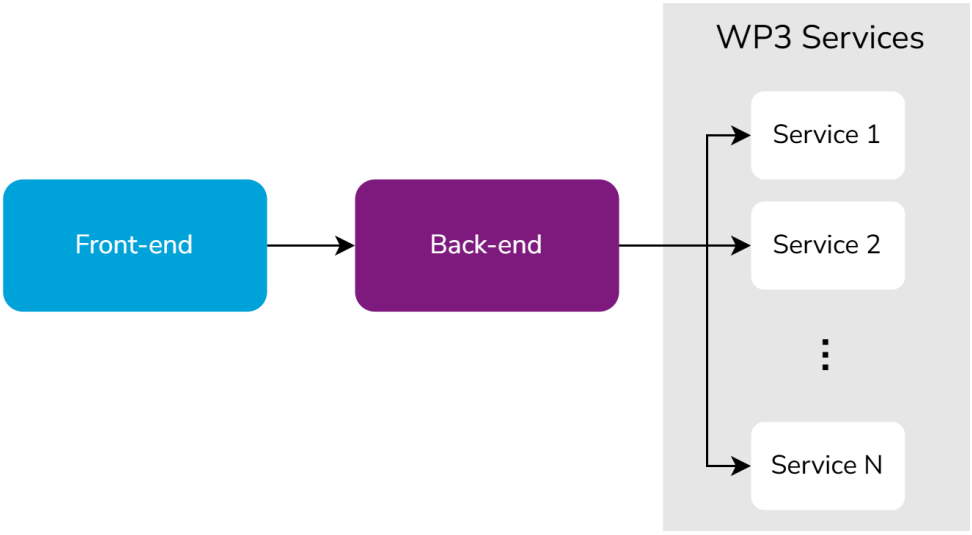

Figure 2 - First version of AI4TRUST platform

Each component adheres to high-security standards, including authentication and encryption, to ensure secure communication between parts, enhance usability, and safeguard user data. For further details on security, please refer to Sections 3.1.1 and 4.1.1.

This first version of the AI4TRUST platform is suitable for enabling the forthcoming pilot evaluation (envisaged in WP6 activities) to highlight the value of the "AI-driven data analysis methods" and show when these could be used in the end-users' workflows, which of these take priority for the end-users and how much these can be considered reliable, helping also the refinement of the individual exploitation plans (envisaged in WP7 activities).

Furthermore, it is worth noting that the "AI-driven data analysis methods" offered in this first version of the AI4TRUST platform will constitute the foundation of the automated pipeline that will power the next releases of the platform. In particular, these tools will enable the "News Items Monitoring Dashboard" of the second version of the AI4TRUST platform (that will be described in the forthcoming "D5.6 - AI4TRUST Platform v2") and of the "Analysis at-scale of collected data" of the final version of the AI4TRUST platform (that will be described in the forthcoming "D5.7 - Final AI4TRUST Platform"). The feedback received from the end-users will thus be essential to

improve the “AI-driven data analysis methods” and consequently the performances of the next releases of the platform.

3. Front-end

The front-end serves as the primary interface for user interactions on the AI4TRUST Platform. Accessible via a web-based UI, this first version offers a user-friendly environment enabling users to seamlessly access and utilise a suite of AI tools designed to tackle disinformation. By adhering to key design principles and employing also a mobile-first approach, the application ensures full responsiveness and compatibility with different devices and display configurations. The platform also offers an additional dark mode option. An example can be seen in Figure 3.

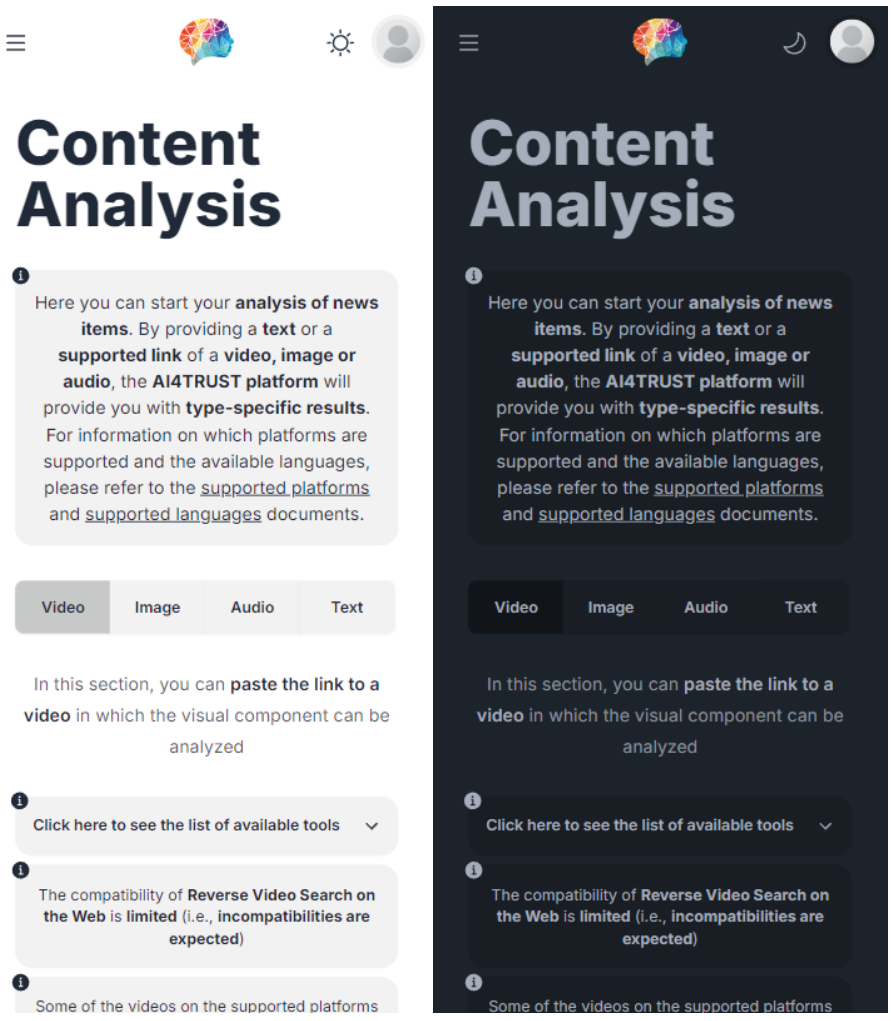

Figure 3 - Responsive design: Light mode (left), Dark mode (right)

As for the structure, it is organised as follows:

- **Home page (Section 3.2):** A landing page containing a brief introduction to the platform;
- **Authentication page (Section 3.2):** Contains a form used for user authentication;
- **Content Analysis (Section 3.3):** Contains an interactive form used as input for the different WP3 services, grouped by content type (i.e., image, video, text, audio);
- **Image/Video/Audio/Text Analysis (Sections 3.3.1, 3.3.2, 3.3.3 and 3.3.4):** The analysis page contains the results of each WP3 service, and differs depending on the content type of the input.

A visual example of the user flow can be seen in Figure 4. Both the home and the authentication pages can be accessed without any kind of authentication (represented in grey), while the analysis interfaces require a username and password (represented in blue). For further details about the authentication process, the security standards used, and the overall methods used to ensure the protection of the platform, please refer to Sections 3.1.1 and 4.1.1.

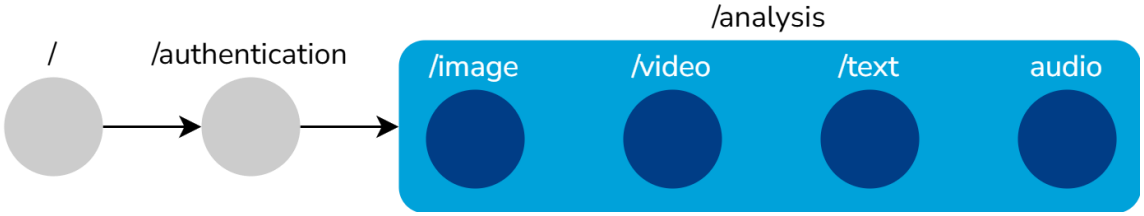

Figure 4 - User interface workflow

3.1. Implementation

The application is built entirely with React[1], utilising TailwindCSS[2] and the DaisyUI[3] component library for design. These tools enhanced code readability and accelerated the development process, while maintaining a simple and intuitive design.

To manage the various packages of the project, Yarn[4] was utilised, effectively mitigating dependency-related issues and environment inconsistencies.

Zustand[5] was utilised for global state management. As a lightweight, performant, and scalable state management library for React, it streamlines the process by enabling the creation of stores with minimal boilerplate code. This approach enhances efficiency in managing and sharing state

[1] https://react.dev/
[2] https://tailwindcss.com/
[3] https://daisyui.com/
[4] https://yarnpkg.com/
[5] https://zustand-demo.pmnd.rs/

across components, without the added complexity associated with more extensive libraries like Redux[6].

To handle the routing process, React Router V6[7] was used, enabling seamless client-side routing and enhancing user experience by allowing dynamic navigation without full page reloads.

For the structure of the front-end development project, a "grouping by file types and features" one was employed. This organisational approach categorises files based on their types, such as components, utilities, and assets, as well as by specific features or functionalities. The advantages of this structure include improved maintainability and scalability, enabling easier location and management of related files. It also enhances collaboration among team members, facilitating a clearer understanding of the project's architecture. By aligning files with their respective features, this structure supports modular development and encourages code reusability, ultimately leading to a more efficient workflow.

An illustration of the front-end structure can be seen in Figure 5.

```
├── public/
├── src/
│   ├── assets/
│   │   └── main.css
│   ├── components/
│   │   ├── authentication/
│   │   │   ├── SubcomponentA.jsx
│   │   │   ├── SubcomponentB.jsx
│   │   │   ├── ...
│   │   │   └── index.jsx
│   │   ├── uploadForm/
│   │   │   ├── SubcomponentA.jsx
│   │   │   ├── SubcomponentB.jsx
│   │   │   ├── ...
│   │   │   └── index.jsx
│   │   ├── ...
│   │   ├── ui/
│   │   │   ├── Dropdown.jsx
│   │   │   ├── Tooltip.jsx
│   │   │   ├── ...
│   │   │   └── index.jsx
│   │   ├── WP3Tools/
│   │   │   ├── Audio/
```

[6] https://redux.js.org/

[7] https://reactrouter.com/

```
│   │   │   │   ├── AudioAnomalyDetection/
│   │   │   │   └── ...
│   │   │   ├── Image/
│   │   │   │   ├── ImageDeepfakeDetection/
│   │   │   │   └── ...
│   │   │   ├── Video/
│   │   │   │   ├── ReverseVideoSearch/
│   │   │   │   └── ...
│   │   │   ├── Text/
│   │   │   │   ├── VerdictGeneration/
│   │   │   │   └── ...
│   │   │   ├── Descriptions/
│   │   │   │   ├── VideoDeeepfakeDescription.jsx
│   │   │   │   ├── AudioTrascriptionDescription.jsx
│   │   │   │   └── ...
│   │   │   └── index.jsx
│   ├── hooks/
│   │   ├── useAuth.js
│   │   ├── useAutoresizeTextArea.js
│   │   └── ...
│   ├── layouts/
│   │   ├── AuthLayout/
│   │   ├── MainLayout/
│   │   ├── ToolLayout/
│   │   └── ...
│   ├── pages/
│   │   ├── AnalysisPage.jsx
│   │   ├── ErrorPage.jsx
│   │   ├── HomePage.jsx
│   │   └── ...
│   ├── store/
│   │   ├── useSharedAnalysisStore.js
│   │   ├── useAuthStore.js
│   │   ├── index.js
│   │   └── ...
│   └── utils/
│       └── Utils.js
├── submodules/
│   └── ai4trust-backend-openapi/...
├── target/...
└── ...
```

Figure 5 - Front-end project structure

The **public/** folder contains all the static assets such as images, favicons, etc.

The main project content lies under the **src/** folder, which contains the following sub-folders:

- **assets**: contains the main assets used by the project codebase. In this case it contains the main css file, which is used as the entry point for importing Tailwind's base styles and to configure additional custom styles.
- **components**: It contains all the component-related code. Each component file is grouped together inside folders. For organisational purposes, WP3 tools are grouped together under a folder. UI/atomic components are grouped under the **ui/** folder.
- **hooks**: The hooks folder contains all the custom hooks for the platform.
- **layouts**: it Contains the layouts of the platform, such as the `MainLayout`, used to define both the navbar and footer, or the `AuthLayout`, used to protect the routes from access without authentication.
- **pages**: This folder contains all the pages of the website, where the components are rendered.
- **store**: This folder contains all the Zustand store hooks for state management.
- **utils**: The utils folder contains all the utility functions (e.g., string-related functions, functions for tailwind colour-class generation, etc.).

The **submodules/** folder is used as a container for the back-end openAPI, which is used to generate the APIs situated in the **target/** folder. For additional information, see Section 4.2

To maintain a clean code, SonarLint[8] was used to enhance code quality, improve maintainability, and ensure adherence to the latest standards and conventions.

3.1.1. Security

The first layer of protection for the platform resides in the authentication process, which requires users to provide a username and a password. Currently, access to the platform is available for consortium partners only, therefore account creation is not available through the platform and is managed manually. The frontend of the platform employs an SSL certificate and API calls are secured via the HTTPS protocol.

Upon successful login, a JWT token[9], and a refresh token is returned; the latter is used to request a new token upon expiration. Both tokens are stored securely (through the **secure**[10] attribute) and are transmitted exclusively to the server in encrypted requests over HTTPS, safeguarding them against man-in-the-middle[11] attacks. Additionally, the refresh token includes the **httpOnly**

[8] https://www.sonarsource.com/products/sonarlint/
[9] https://jwt.io/
[10] https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#security
[11] https://developer.mozilla.org/en-US/docs/Glossary/MitM

attribute, preventing modification through JavaScript, thus reducing the risk of cross-site scripting[12] attacks.

Each protected call checks for the validity of the JWT (if it is expired or forged). In case the given JWT is expired, it asks for a new JWT using the refresh token, and it is updated consequently. If the JWT check is not valid due to manual alterations (injection of fake/aterated tokens), it logs out the user.

3.2. Platform Entry and Access Points

The entry point of the platform, shown in Figure 6, features a brief welcome message.

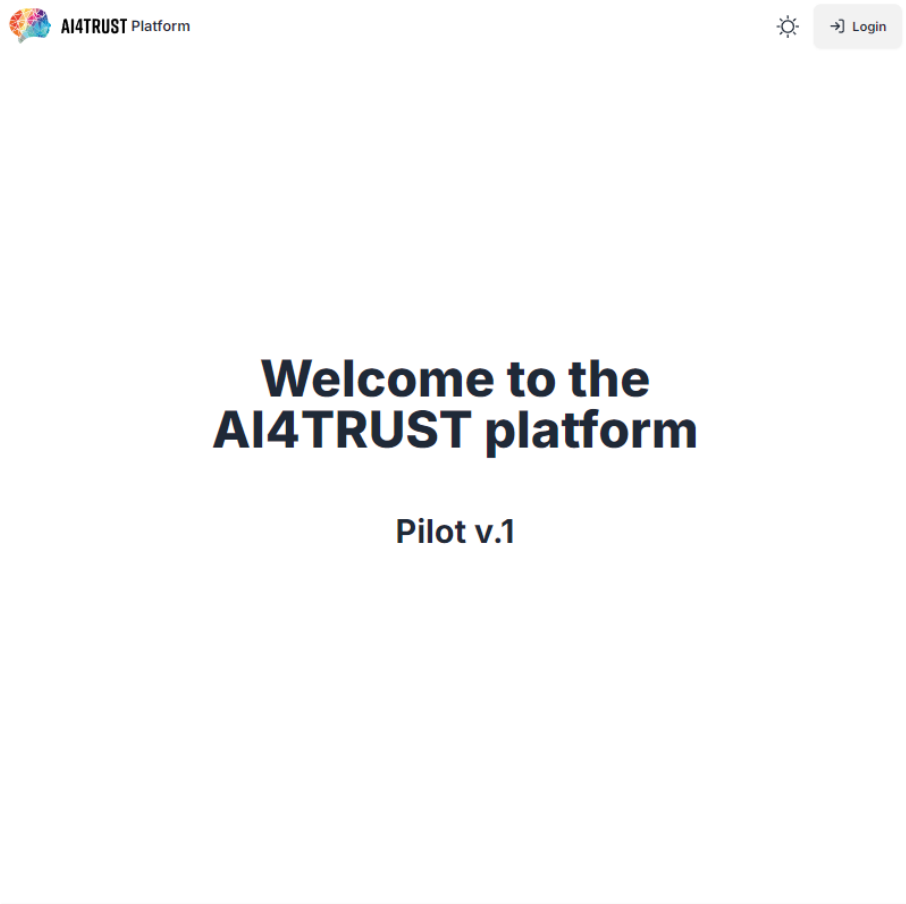

Figure 6 - Home page

[12] https://developer.mozilla.org/en-US/docs/Web/Security/Types_of_attacks#cross-site_scripting_xss

The access point (see Figure 7) features a login form for user authentication. React Hook Form[13] is used to implement a performant and robust schema validation, ensuring that the authentication process maintains high security standards. For further details about the authentication process, the security standards used, and the overall methods used to ensure the protection of the platform, please refer to the respective sections 3.3.1 and 4.4.1.

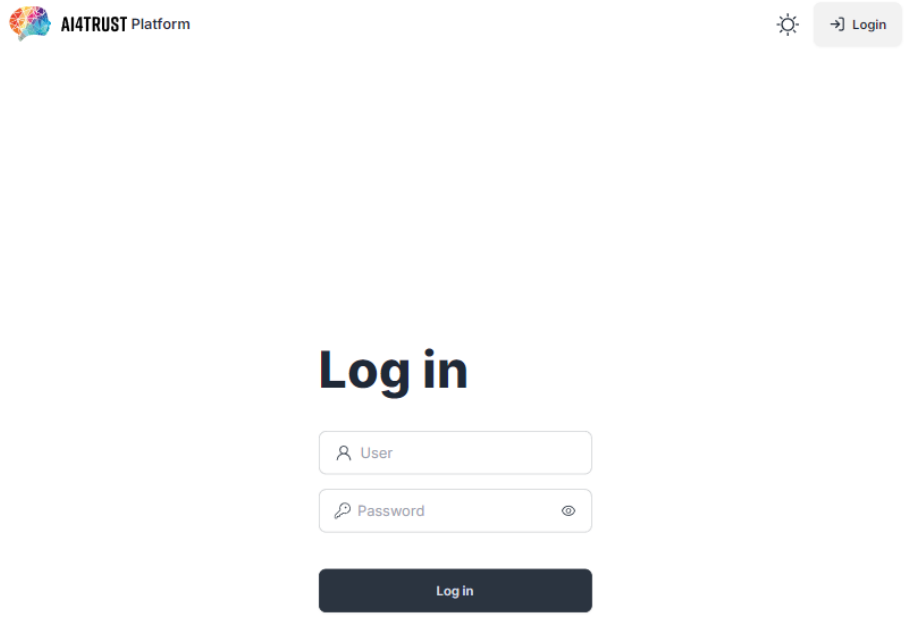

Figure 7 - Login page

3.3. Content Analysis

The content analysis page serves as an entrypoint for providing content input for the analysis step. It includes a selector based on the type of input to be analysed (see Figure 8).

[13] https://react-hook-form.com/

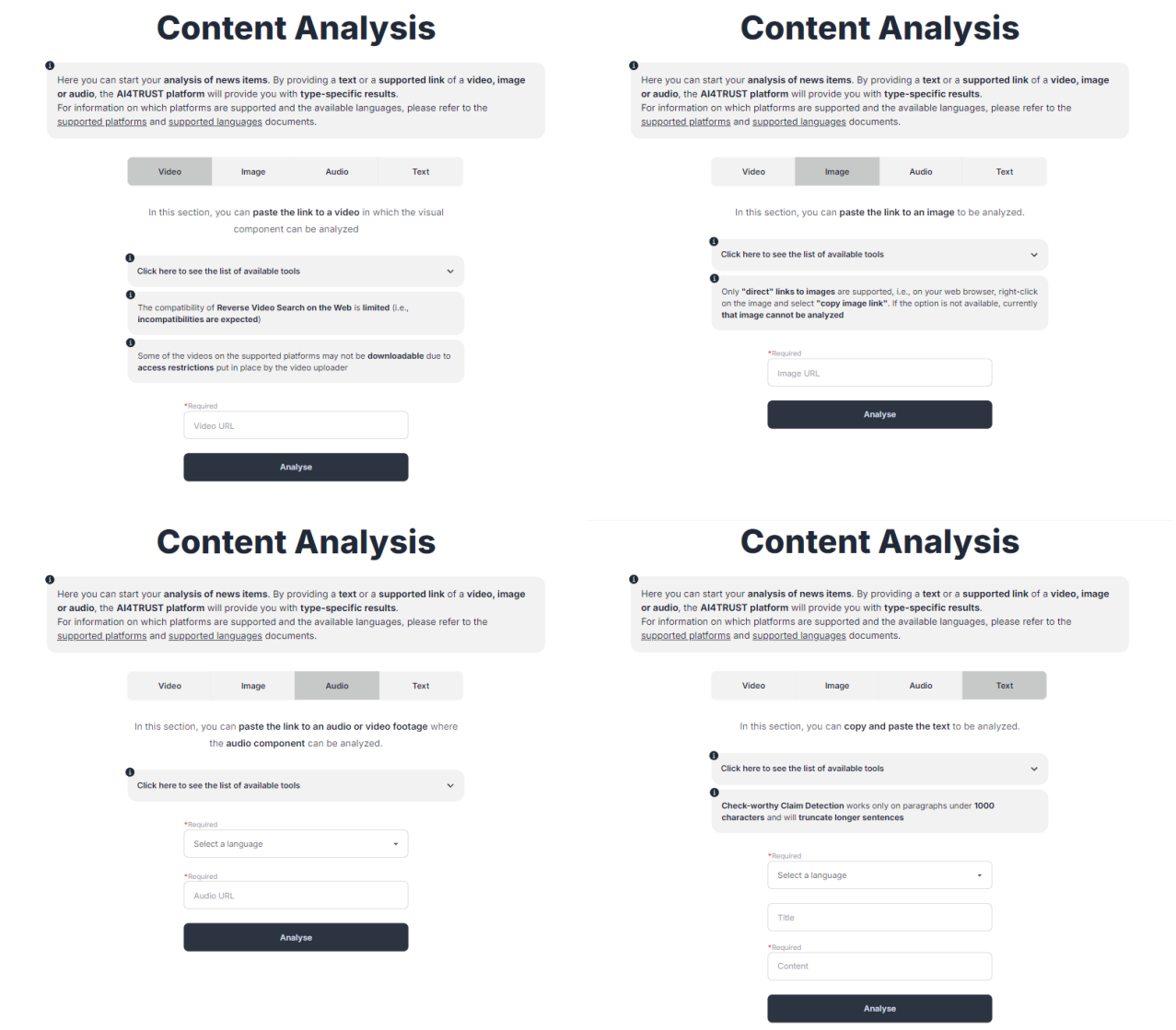

Figure 8 - Upload page UI for Video (upper-left), Image (upper-right), Audio (lower-left) and Text (lower-right) inputs

Video, audio and image inputs require the content URL; for text inputs, the platform accepts the textual content and an optional title.

Additionally, both audio and text inputs require a language to be selected from a list of available languages (see Table 1).

Category	Tool	English (en)	Greek (el)	Romanian (ro)	Italian (it)	Spanish (es)	Polish (pl)	German (de)	French (fr)
Image Analysis	**AI-generated Image Detection**	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic
Video Analysis	**Reverse video search on the Web**	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic

Category	Tool	English (en)	Greek (el)	Romanian (ro)	Italian (it)	Spanish (es)	Polish (pl)	German (de)	French (fr)
	Video DeepFake Detection	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic	Language Agnostic
Audio Analysis	**Audio Transcription**	yes	no	yes	yes	yes	yes	yes	yes
	Audio Deepfake Detection	yes	not optimised	not optimised	not optimised	not optimised	not optimised	not optimised	not optimised
	Audio Anomaly Detection	yes	not optimised	not optimised	not optimised	not optimised	not optimised	not optimised	not optimised
Text Analysis	**Disinformation Signals Detection**	yes	yes	yes	yes	yes	yes	yes	yes
	Check-worthy Claim Detection	yes	no	no	yes	yes	no	no	no
	Verdict Generation	yes	no	no	no	no	no	no	no

Table 1 - List of supported languages

Similarly to the login page, each form on the upload page undergoes schema validation to ensure the accuracy of the initial input before sending it to the platform back-end (see Chapter 4).

The analysis page for image, video, audio, and text is structured into two sections:

- **Preview:** The preview tab (situated on the left) contains the preview for the input content;
- **Analysis:** The analysis tab (situated on the right) presents the analysis result of each WP3 tool.

Both the preview and analysis tabs are customised for each type of input, as detailed in the following subsections. The preview tab is fixed, allowing the user to keep the raw content in view while scrolling through the analysis tab to interact with each tool. This setup ensures that users can continuously reference the original content as they explore the analysis results.

Each tool includes a detailed explanation of its purpose and guidance on how to interpret the results and is equipped with a timer that tracks its execution time, offering feedback to the users about the current status of the analysis.

3.3.1. Video

The video preview features a video player[14] compatible with multiple sources (e.g., YouTube, Facebook, Twitch, SoundCloud, Streamable, Vimeo, Wistia and DailyMotion). If the source is unavailable or not compatible, a placeholder and a link to the content are provided. An example of both the available and unavailable video previews can be seen in Figure 9.

Figure 9 - Thumbnail preview: Compatible source preview (left), Unavailable/not compatible source preview (right)

The video analysis shows the results for the **video deepfake detection** tool, and the **reverse video search**.

The video deepfake detection tool employs a set of AI detectors to analyse whether a person's face has been manipulated using techniques such as Face Swap or Face Reenactment.

The video deepfake detection tool UI presents a label indicating the final video analysis (see Figure 10). If at least one detector identifies segments of the video with manipulated content, the video is marked as suspicious. Conversely, if no manipulations are detected by either model, the video is considered either real or potentially manipulated using a deepfake method other than face swap or reenactment.

[14] React Player: https://github.com/cookpete/react-player?tab=readme-ov-file

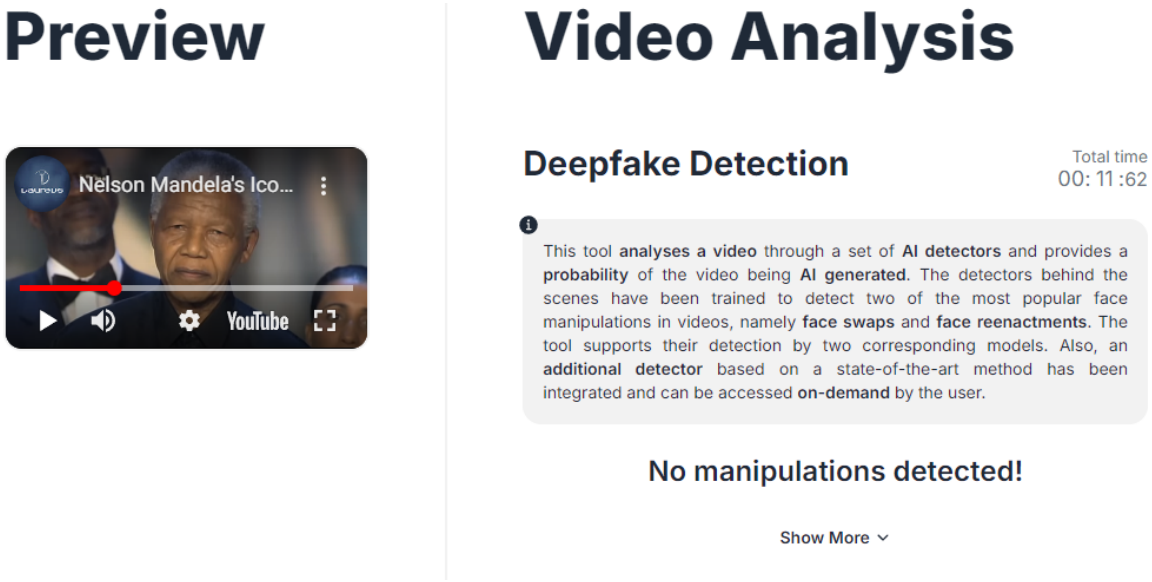

Figure 10 - Video Deepfake DetectionThumbnail preview

By clicking the "Show More" button, users can access a more detailed analysis of the results (see Figure 11). This expanded view provides, for each model: a description of the model, the probability that the video contains a manipulation tackled by that specific model, and a textual explanation of the probability.

For benchmarking purposes, the platform also includes a state-of-the-art model known as the Temporal Incoherence Checker[15], which can be run on-demand by the user. This model is included for reference only and is not part of the AI4TRUST project.

[15] https://ieeexplore.ieee.org/document/9710282

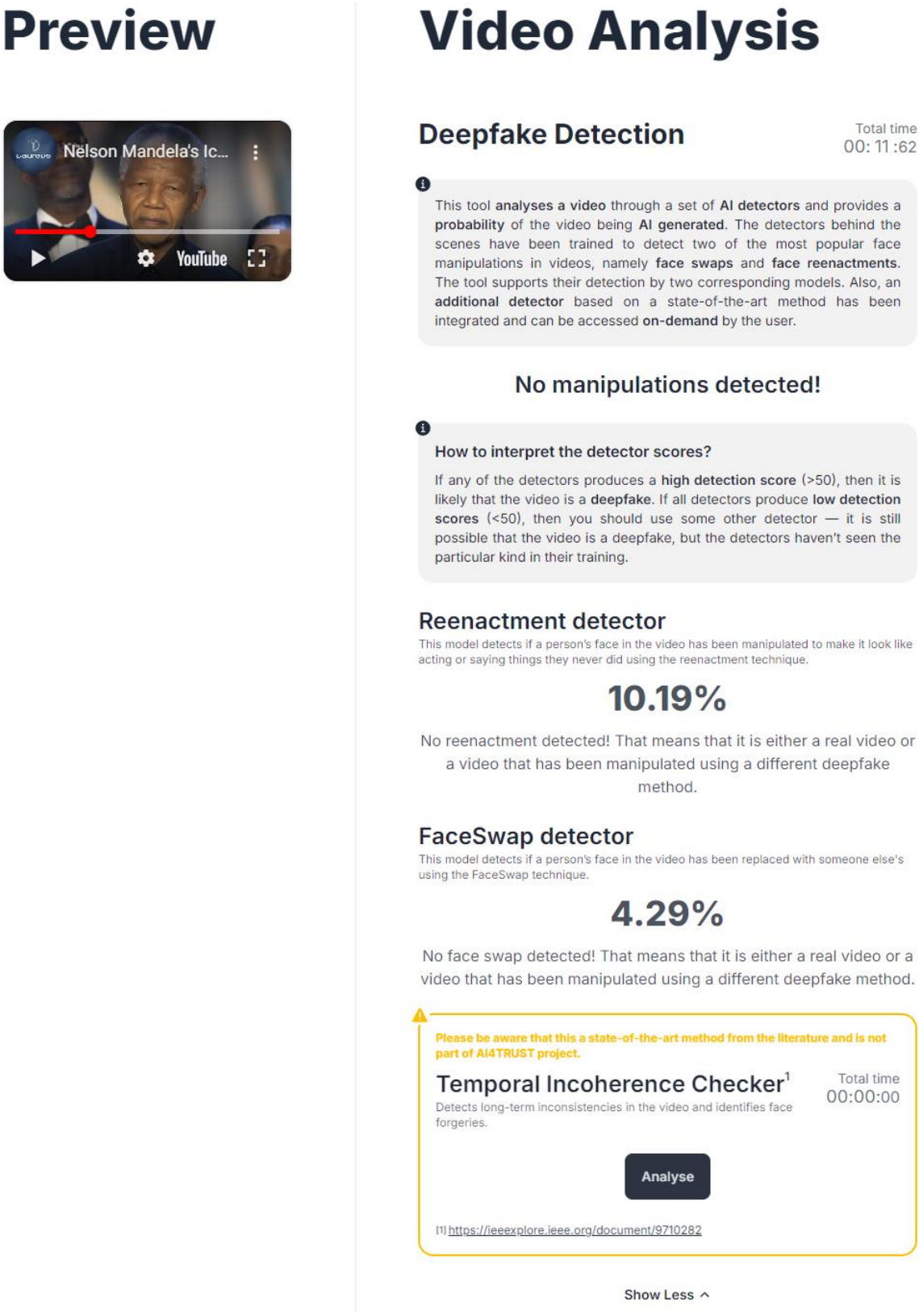

Figure 11 - Video Deepfake Detection expanded

The **reverse video search tool** works by identifying near-duplicate videos. It extracts a set of representative keyframes from the video and conducts an automated search to find visually similar content or matching videos. This helps detect whether content has been reused or modified across different sources. The interface features multiple tabs, each dedicated to a specific function. As shown in Figure 12, there are two main tabs: "Similar Videos", which displays the near-duplicate videos found on the web, and "More Online Sources", which presents visually similar additional content.

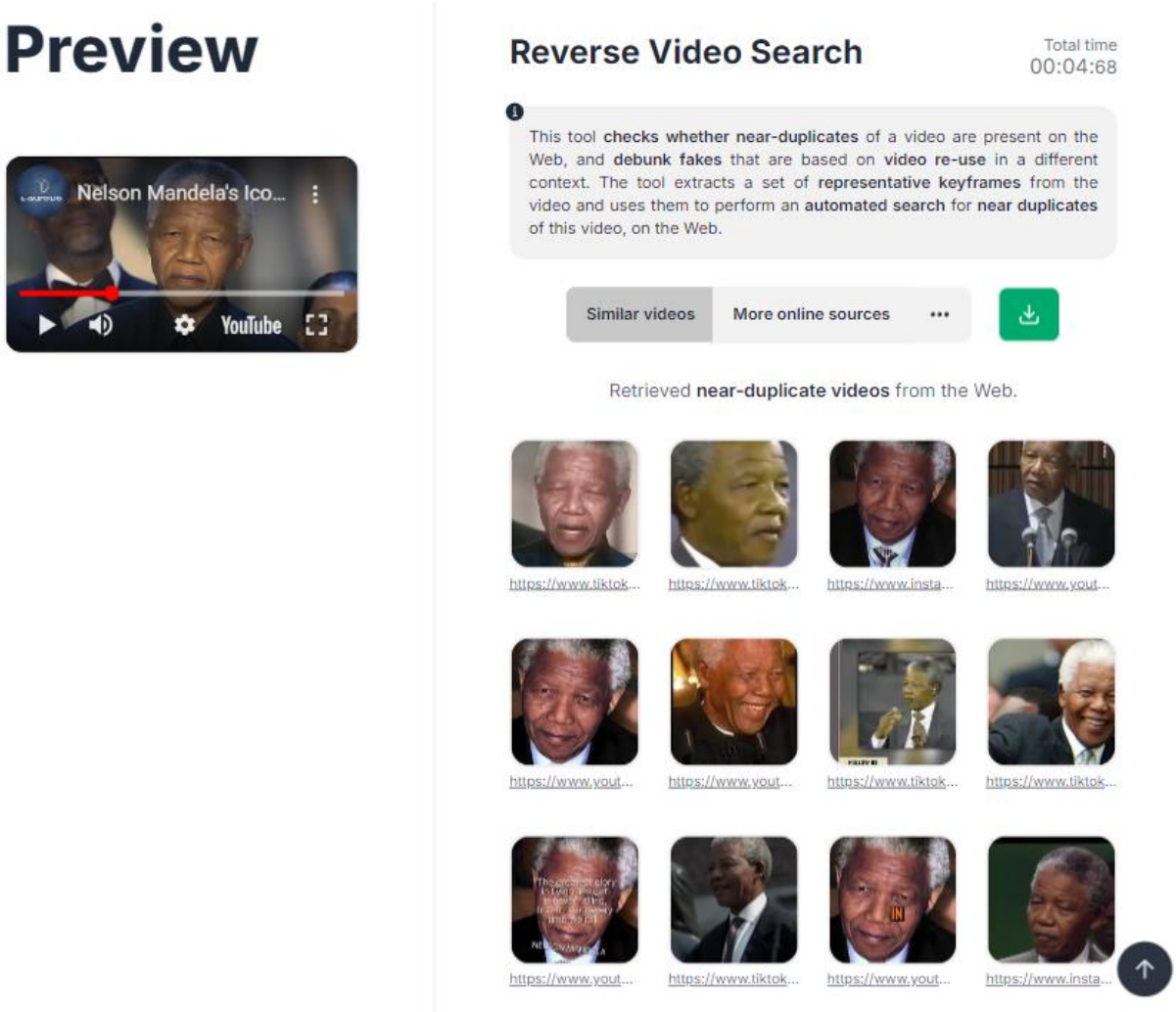

Figure 12- Reverse Video Search: Similar videos tab selected

The tool also offers an explainability feature accessible via an additional tab labelled "List of representative keyframes" (see Figure 13), which can be found under the three dots menu. This tab displays the frames that were extracted and utilised for the automated search, providing insight into the search process.

If a more detailed reverse video search is desired, users can access the “Search additional keyframes” tab, where a list of additional extracted keyframes is available. By clicking on a keyframe, the user initiates a Google Lens reverse video search, allowing for a more thorough examination of similar content.

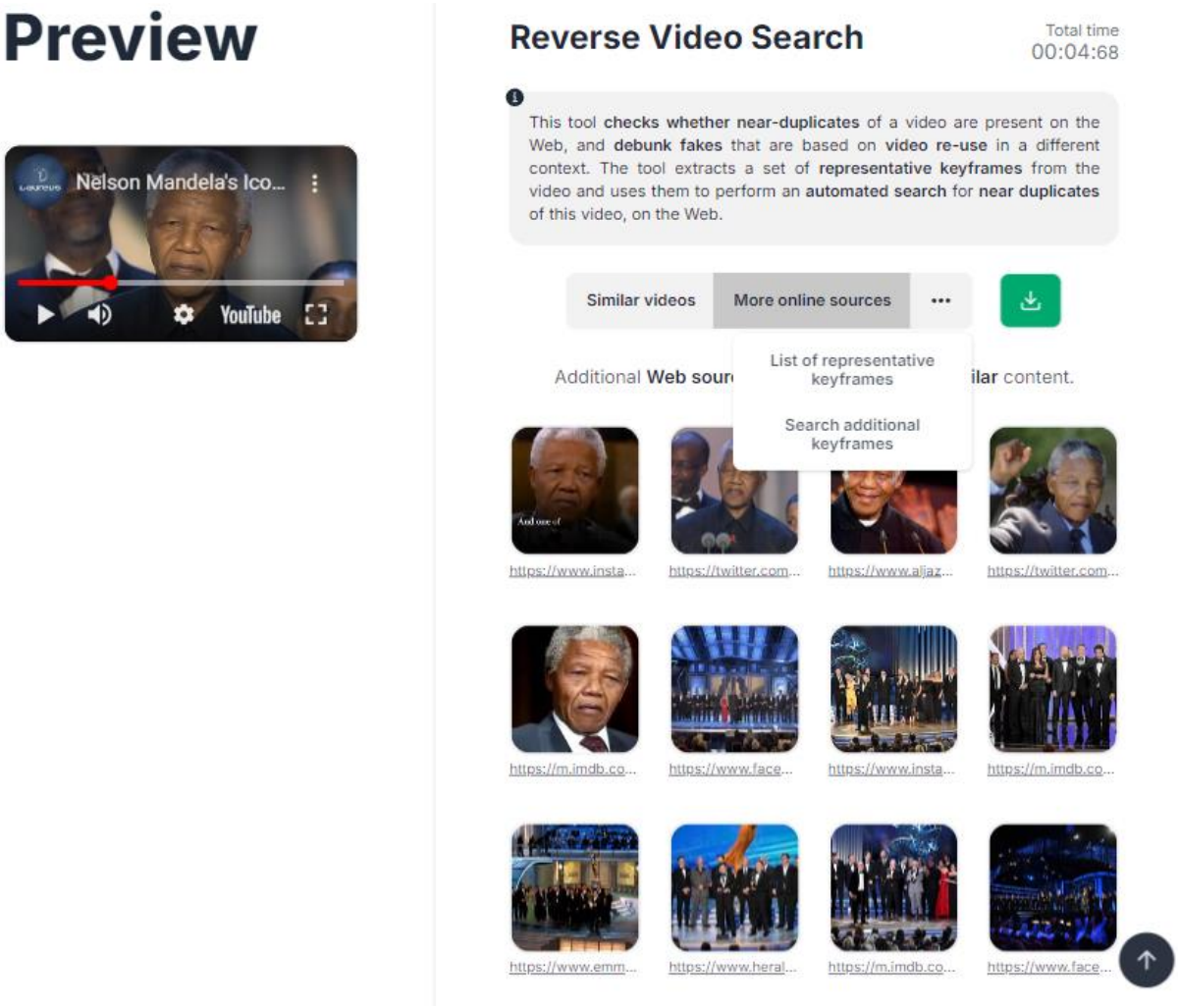

Figure 13 - Reverse Video Search additional tabs

Additionally, it provides options to download both the representative keyframes and the additional extracted keyframes, through the download button.

3.3.2. Image

The image analysis presents the results for the AI-generated image detection tool, which employs a set of AI detectors to check whether an image was generated using different generative models like Diffusion Models (DM)[16] or Generative Adversarial Networks (GAN)[17].

As shown in Figure 14, the initial result is presented as a label indicating one of three outcomes:

- **suspicious**: meaning at least one detector found the image likely to have been generated using a known model;
- **inconclusive**: meaning no detector found sufficient evidence to classify the image as AI-generated;
- **not detected as synthetic**: indicating that the image is either real or generated by a model unknown to the detector.

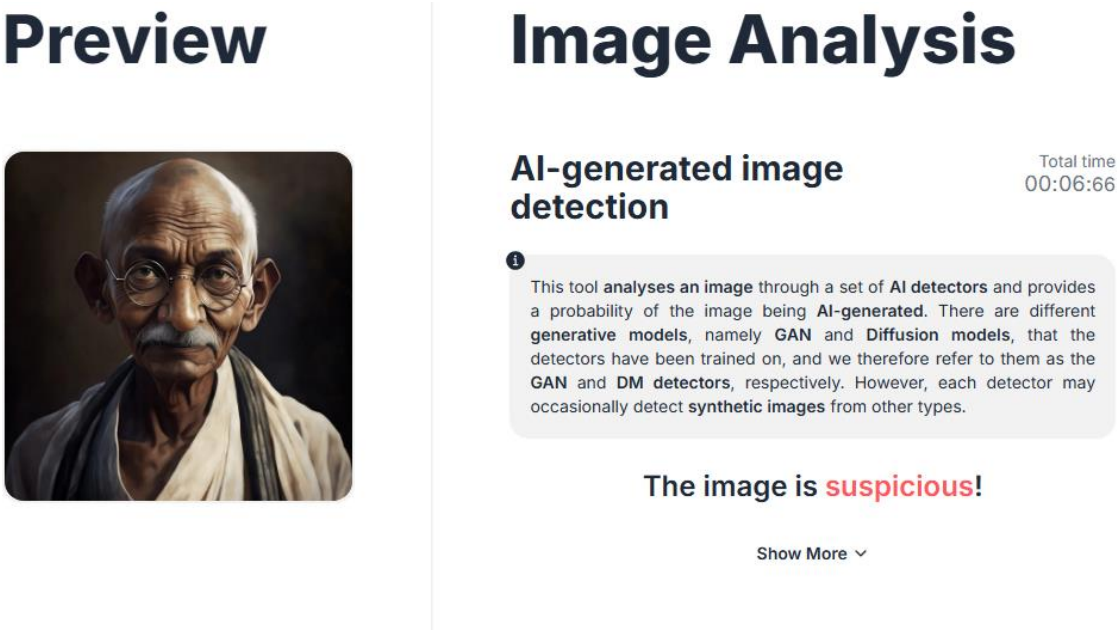

Figure 14 - AI-generated image detection

By clicking the "Show More" button, users can access a more detailed analysis of the results (see Figure 15). This expanded view provides, for each model: a description of the model, the probability that the image was generated by that specific model, and a textual explanation of the probability, offering deeper insights into the image's authenticity.

[16] https://en.wikipedia.org/wiki/Diffusion_model

[17] https://en.wikipedia.org/wiki/Generative_adversarial_network

Preview

Image Analysis

AI-generated image detection

Total time
00:06:66

This tool **analyses an image** through a set of **AI detectors** and provides a probability of the image being **AI-generated**. There are different **generative models**, namely **GAN** and **Diffusion models**, that the detectors have been trained on, and we therefore refer to them as the **GAN** and **DM detectors**, respectively. However, each detector may occasionally detect **synthetic images** from other types.

The image is suspicious!

How to interpret the two detector scores?

If any of the two detectors produces a **high detection score**, then it is likely that the image is **AI-generated** and there is reason to be **suspicious** of it. If one or both of the detectors produce an **"intermediate" score**, the result should be considered **inconclusive**, but note that there is still reason to be **suspicious** of the image.
If both detectors produce **low detection scores**, then our models declare a **non-detection** and you should use another detector. It is still possible that the image is **AI-generated**, but our detectors haven't seen the particular **generative model** in their training.

GAN Detector

This model has been trained to detect images generated through AI techniques known as Generative Adversarial Networks (GAN). Some popular GAN-based generators are ProGAN, StyleGAN, BigGAN, and CycleGAN. Occasionally, it may detect images generated by Diffusion Models.

100.00%

Suspicious! This image is likely generated using a Generative Adversarial Network (GAN). The content depicted in the image probably doesn't exist in real life.

DM Detector

This model has been trained to detect images generated through AI techniques known as Diffusion Models (DM). Some popular DM-based generators are Latent Diffusion, Guided Diffusion, and Glide. In addition it performs reasonably well on images generated by commercial tools such as DALL-E (works well for versions 1 and 2, we have noticed failure in version 3), Firefly, and Midjourney (tested in version 5). Occasionally, it may detect images generated by GANs.

61.47%

Not detected as synthetic by this model! That means that it is either a real image or an image that has been generated by a model that is unknown to this detector.

Show Less ^

Figure 15 - Expanded AI-generated Image Detection

The image preview features a clickable thumbnail that expands to show the full image, as illustrated in Figure 16.

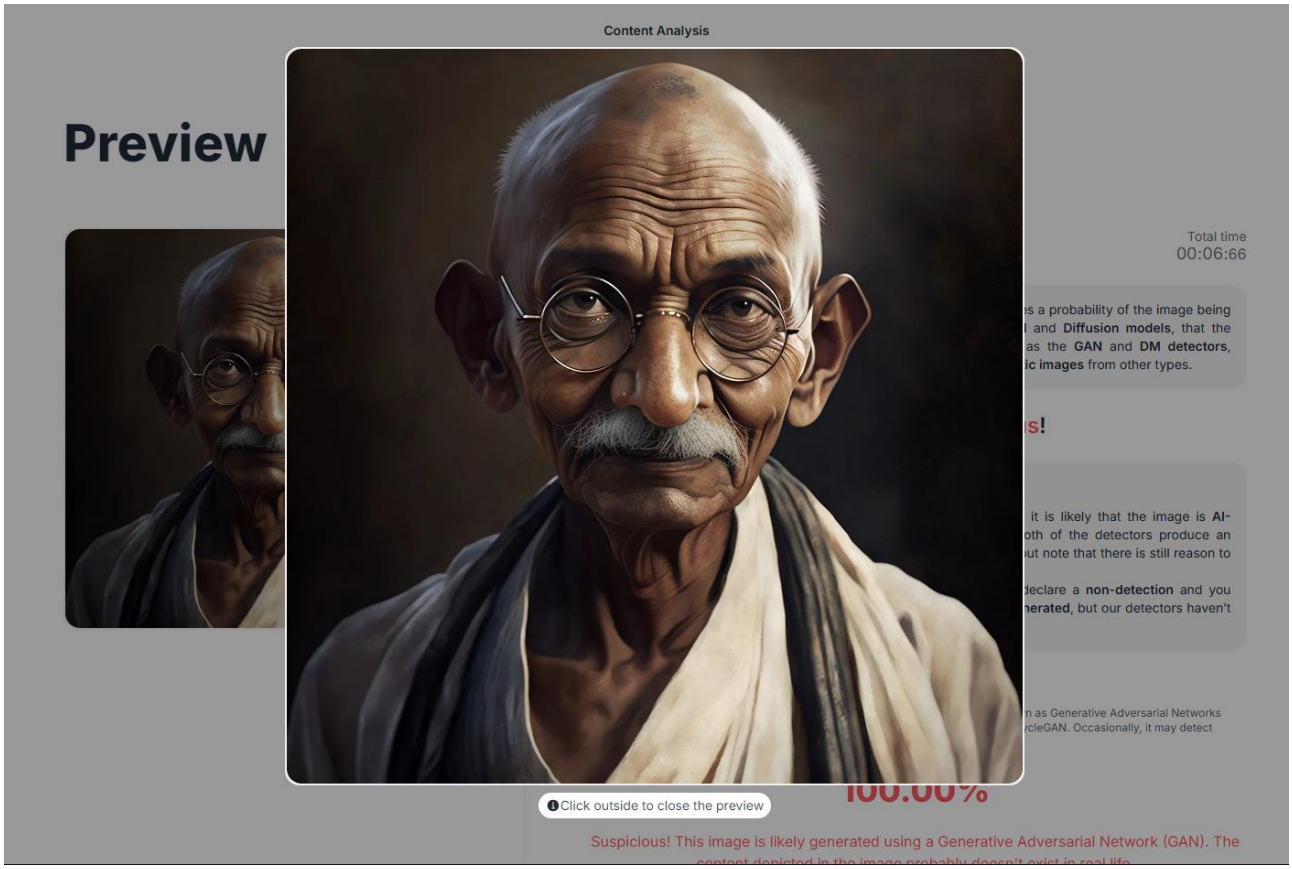

Figure 16 - Image preview

3.3.3. Audio

The audio preview contains a video player analogous to the video preview. The audio analysis contains three tools: **deepfake detection, audio anomaly detection** and **transcription** (speech-to-text).

The deepfake detection tool analyses the entire audio and indicates, with a probability, if it is likely to be real or fully generated using AI tools (see Figure 17).

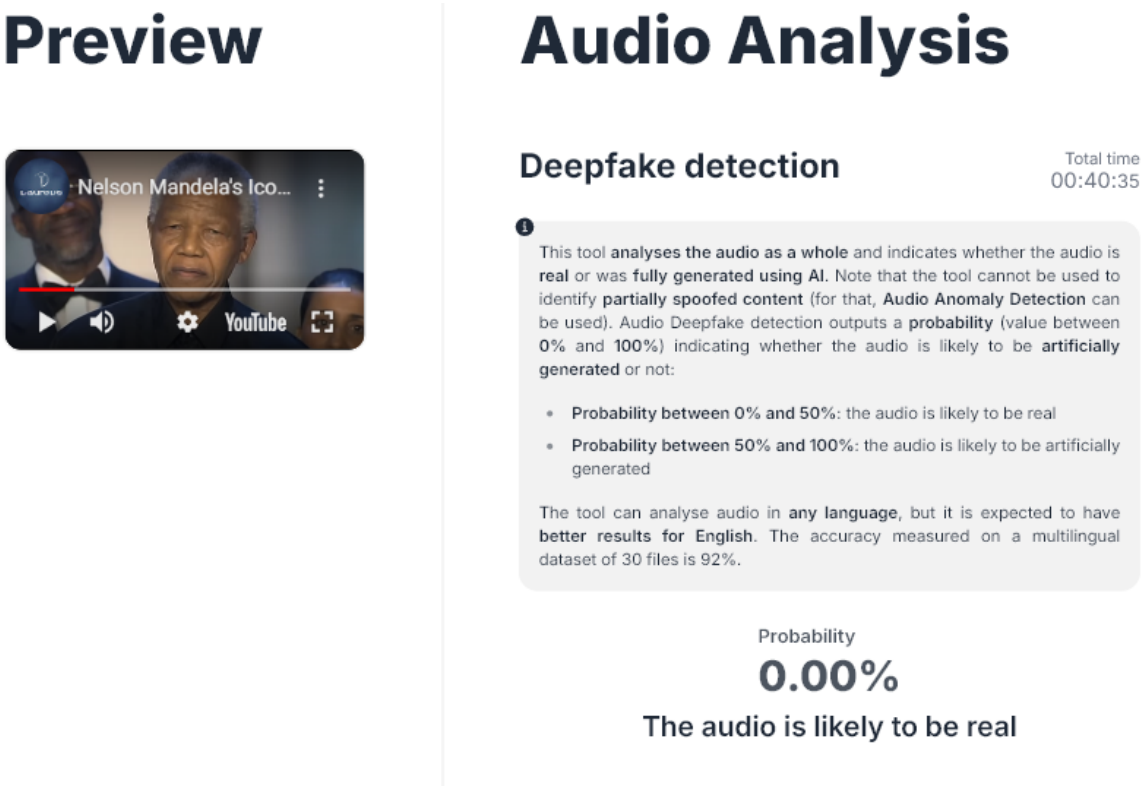

Figure 17 - Audio Deepfake detection

The audio anomaly detection tool (see Figure 18) segments audio into intervals of one-tenth of a second, assigning a probability from 0 to 100% for each segment to indicate its likelihood of being anomalous (i.e., containing splicing points or being AI-generated). It displays a graph illustrating time against probability, featuring a 50% threshold, along with basic statistics that indicate the percentage of anomalous versus bonafide segments in the audio. Selecting a point in the graph sets the video preview to the corresponding timestamp.

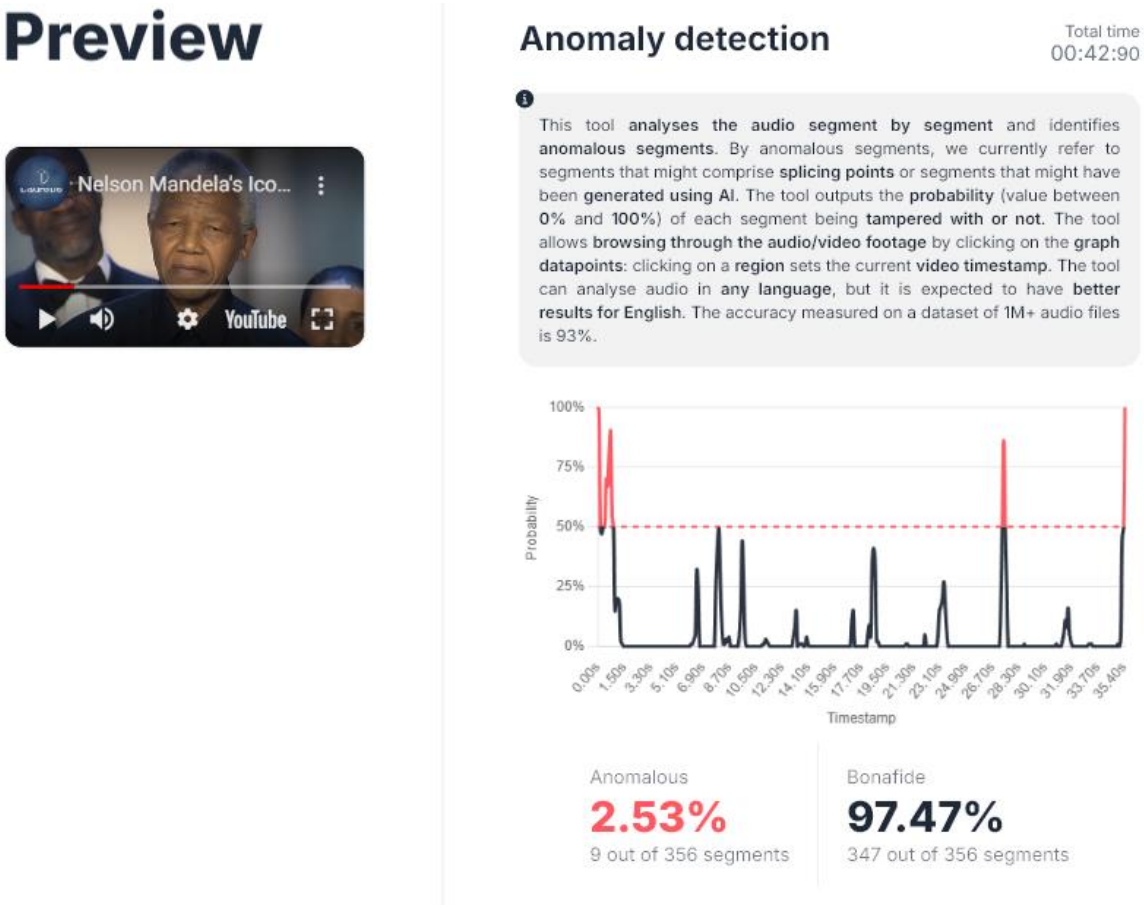

Figure 18 - Audio Anomaly detection

The transcription feature generates a text grouped by speaker, with an option to display the confidence level for each word through a checkbox. Words are dynamically highlighted, and their confidence level can be viewed by hovering over each word. Selecting a word within the transcription syncs the video preview to the corresponding timestamp. Textual analysis can be performed directly by selecting a portion of the transcription and clicking on the “Analyse selected text” button, as shown in Figure 19.

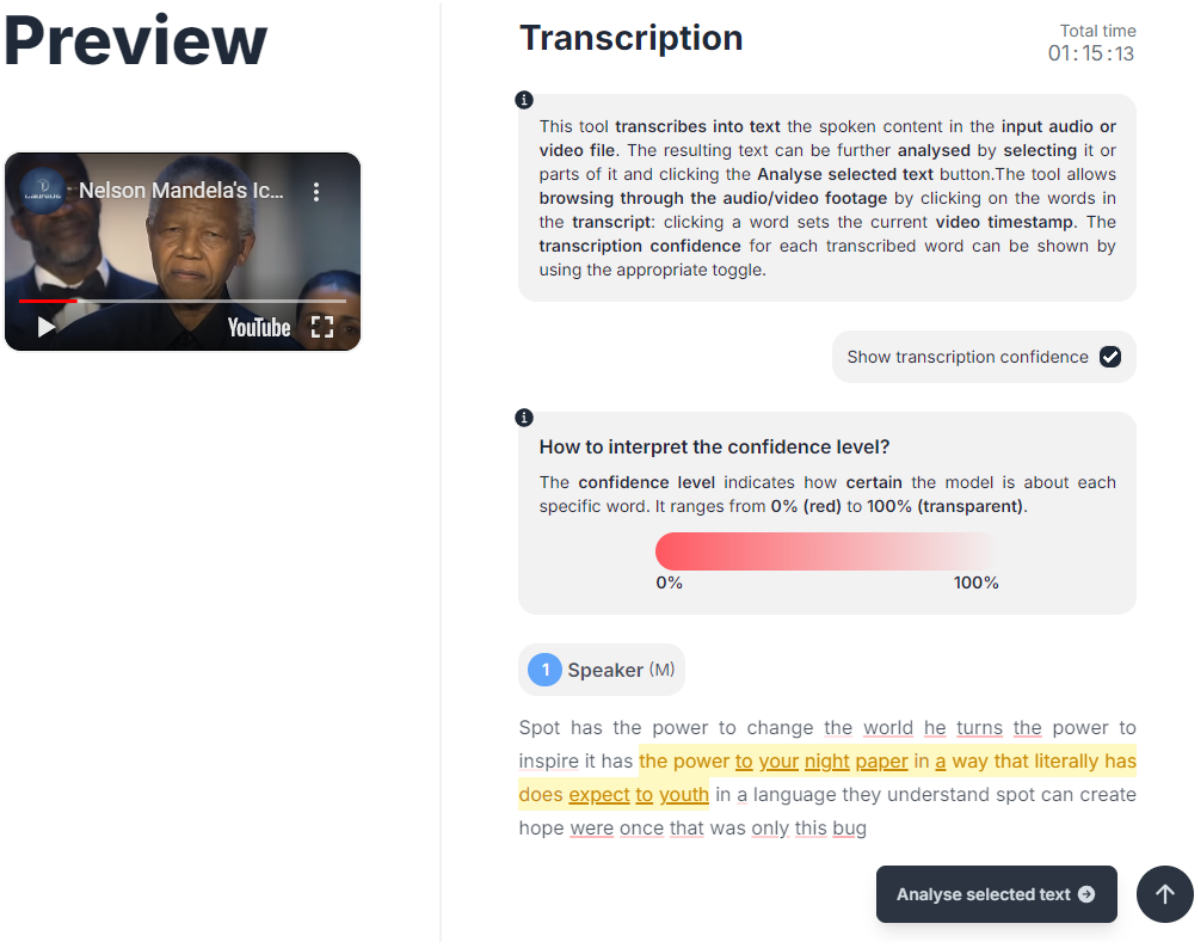

Figure 19 - Transcription tool (selection for textual analysis highlighted in yellow)

3.3.4. Text

The text analysis page offers a preview similar to the image preview, where a portion of the text is displayed and can be fully viewed upon clicking (see Figure 20).

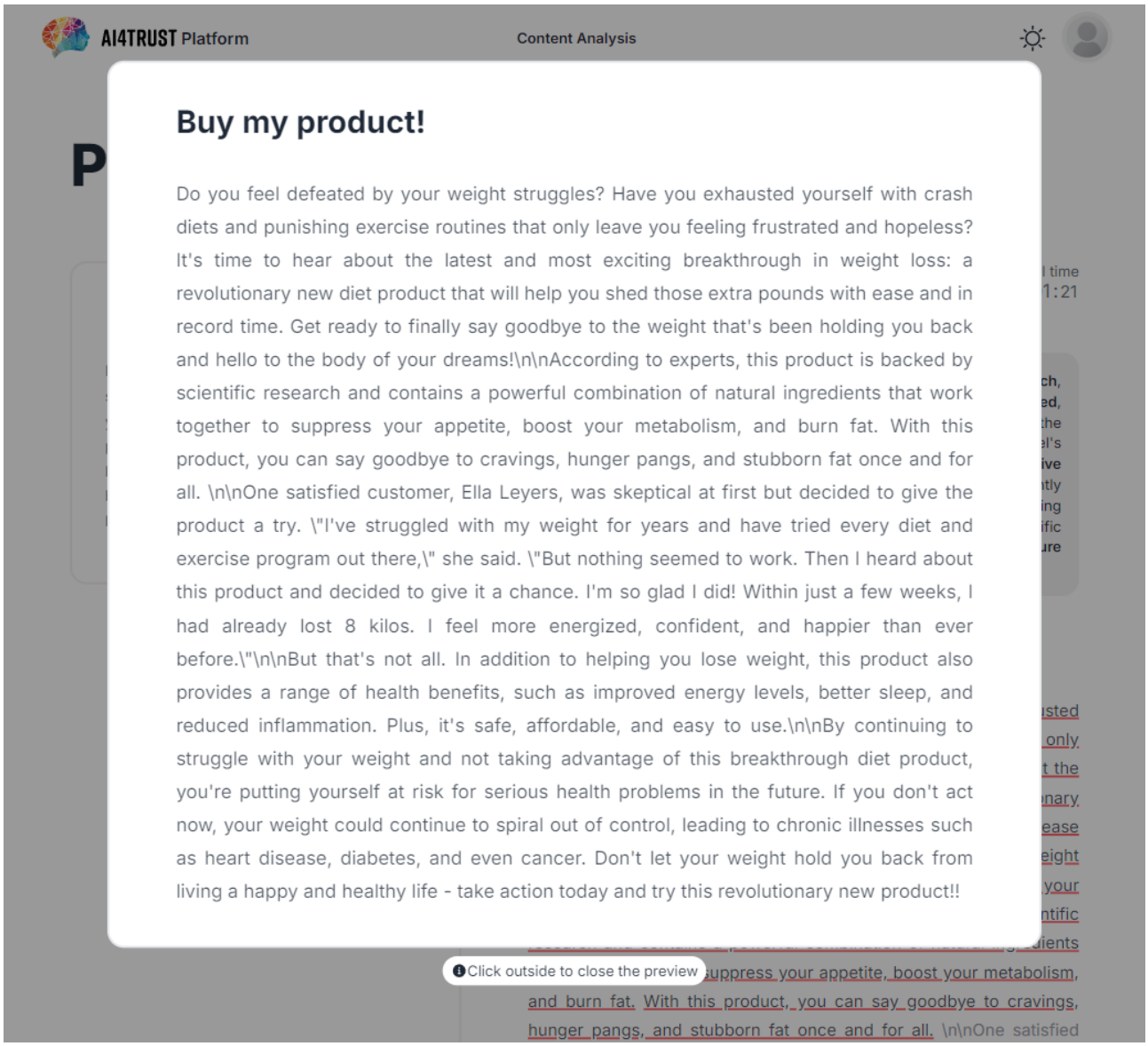

Figure 20 - Text preview

The analysis section includes 3 tools: **disinformation signals detection**, **check-worthy claim detection** and **verdict generation**.

The disinformation signals detection tool analyses text content, assigning labels and corresponding confidence levels to each sentence. Labels include hate speech, offensive language, and clickbait, providing insights into the nature of the content. Users can select a specific label from the available tab, which highlights the relevant sections of the text (see Figure 21). By hovering over a highlighted word, the tool displays the confidence percentage, allowing users to assess the likelihood of each label being accurate.

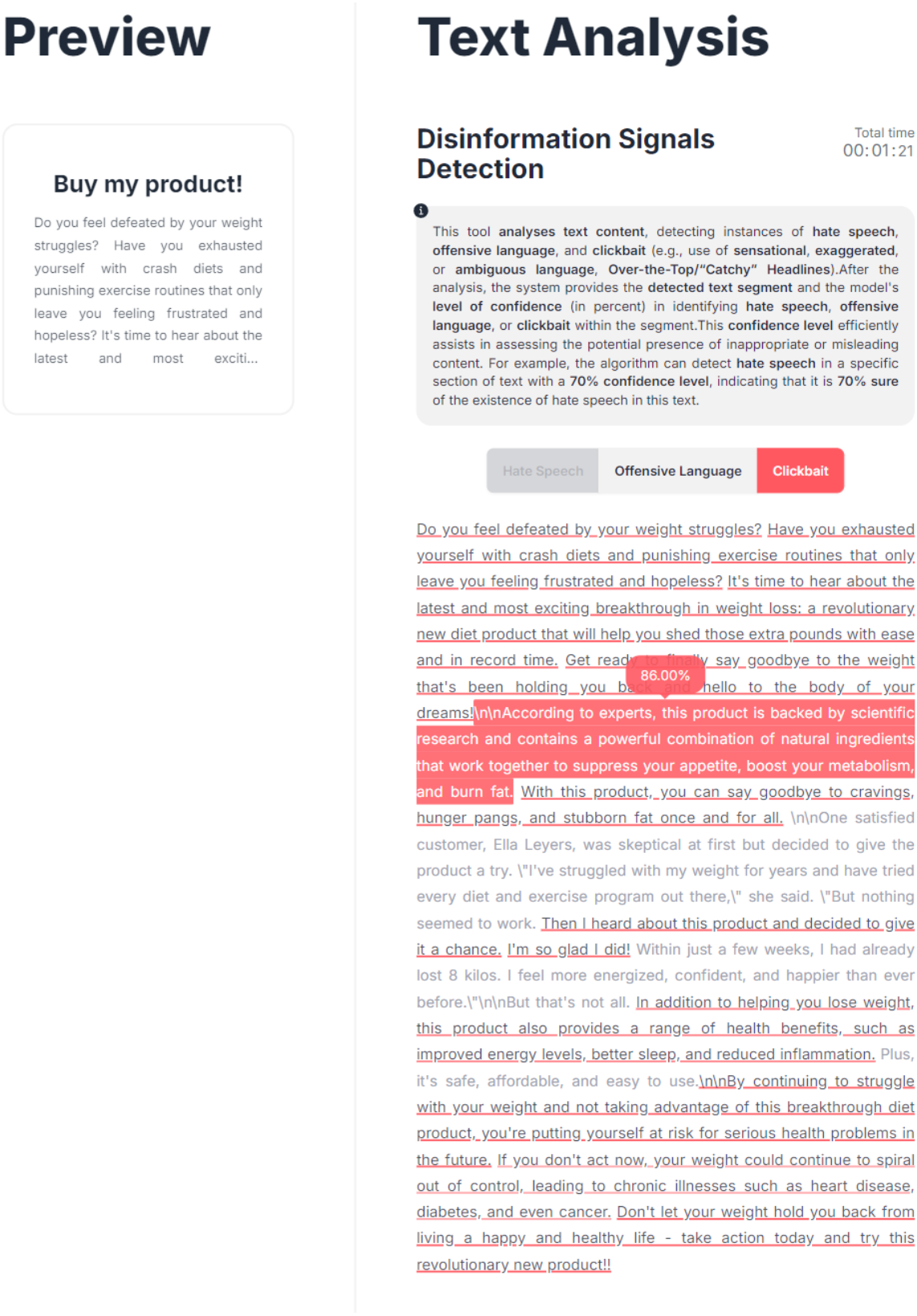

Figure 21 - Disinformation signals detection

The check-worthy claim detection labels content as either check-worthy or not, and provides a score (or confidence), indicating the likelihood of that content being check-worthy (see Figure 22). The criteria for determining check-worthy texts include appearing false, being of public interest, impacting the public, or potentially causing harm. Such texts must contain factual claims. The tool marks as non-check-worthy texts that are opinion-based or consist of easily verifiable facts (e.g., “The sky is blue”).

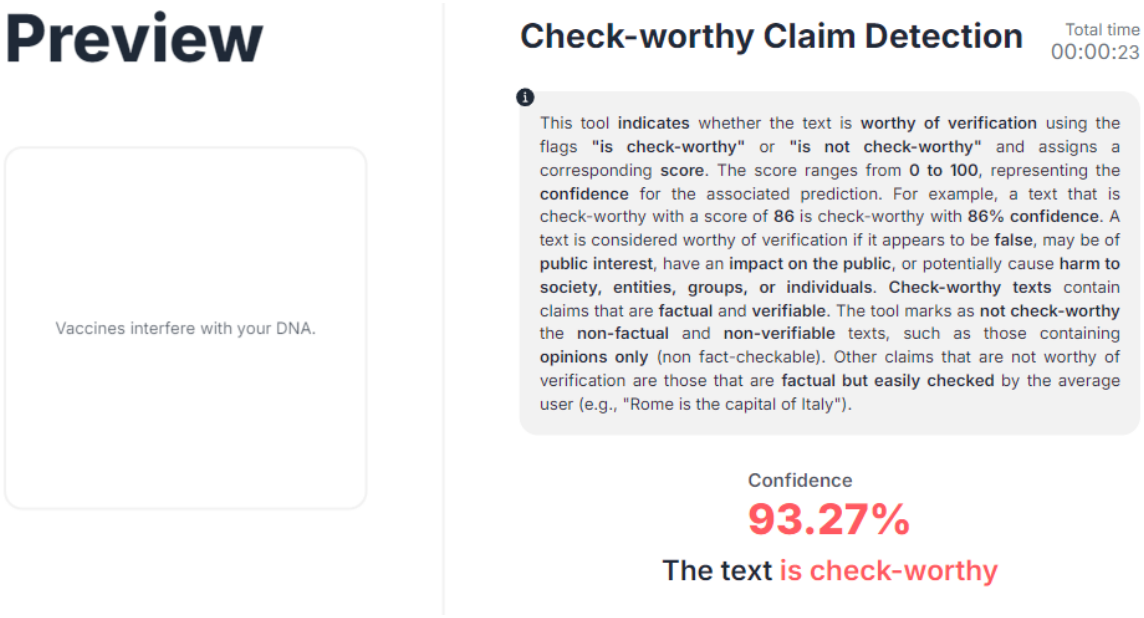

Figure 22 - Check-worthy claim detection

Verdict generation tool requires user input (see Figure 23), including a fact-checking source, a result style in which the verdict will be written (social media or journalistic) and the number of relevant sentences to consider in support of the verdict.

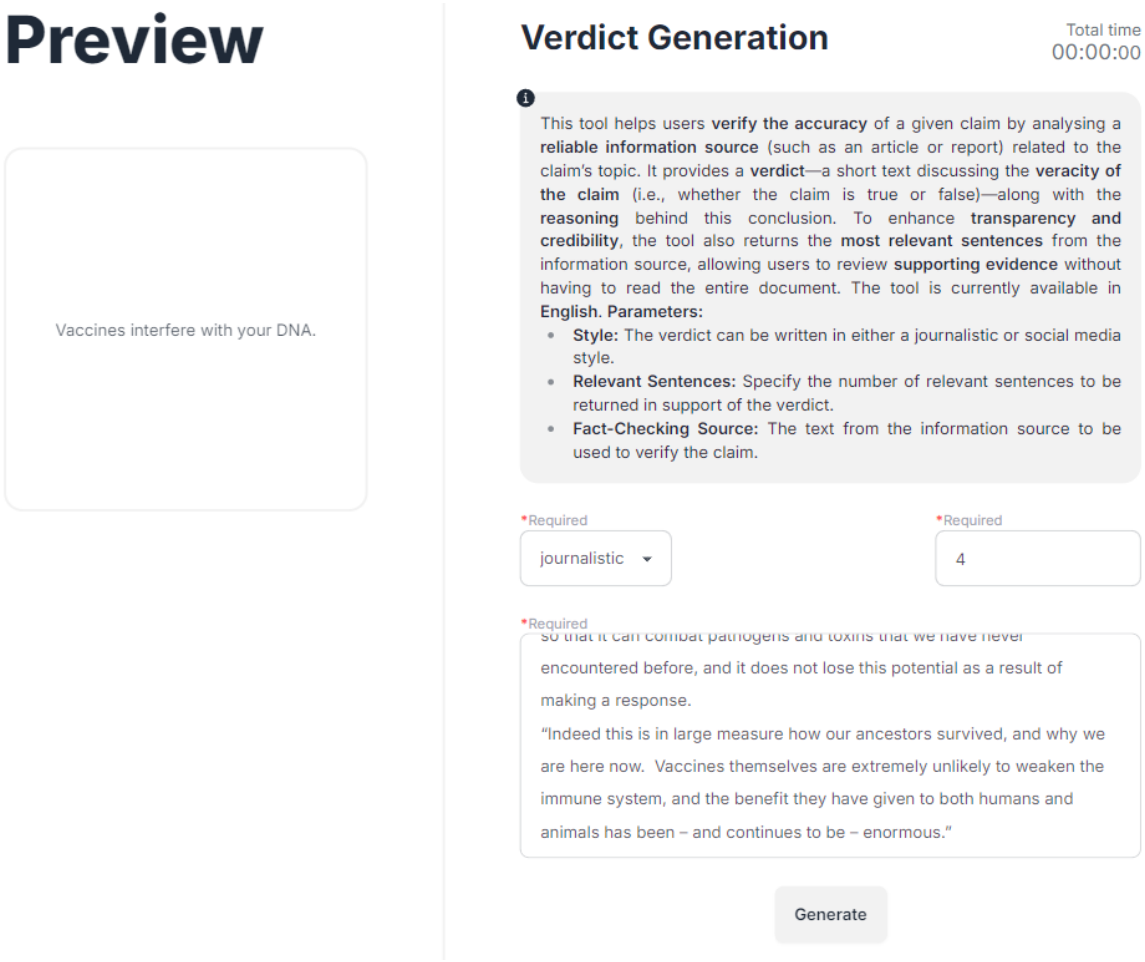

Figure 23 - Verdict generation tool form

Once the input is given, the tool outputs the relevant sentences inside the fact-checking source, and the final verdict. Both the input form and the results can be seen in Figure 24.

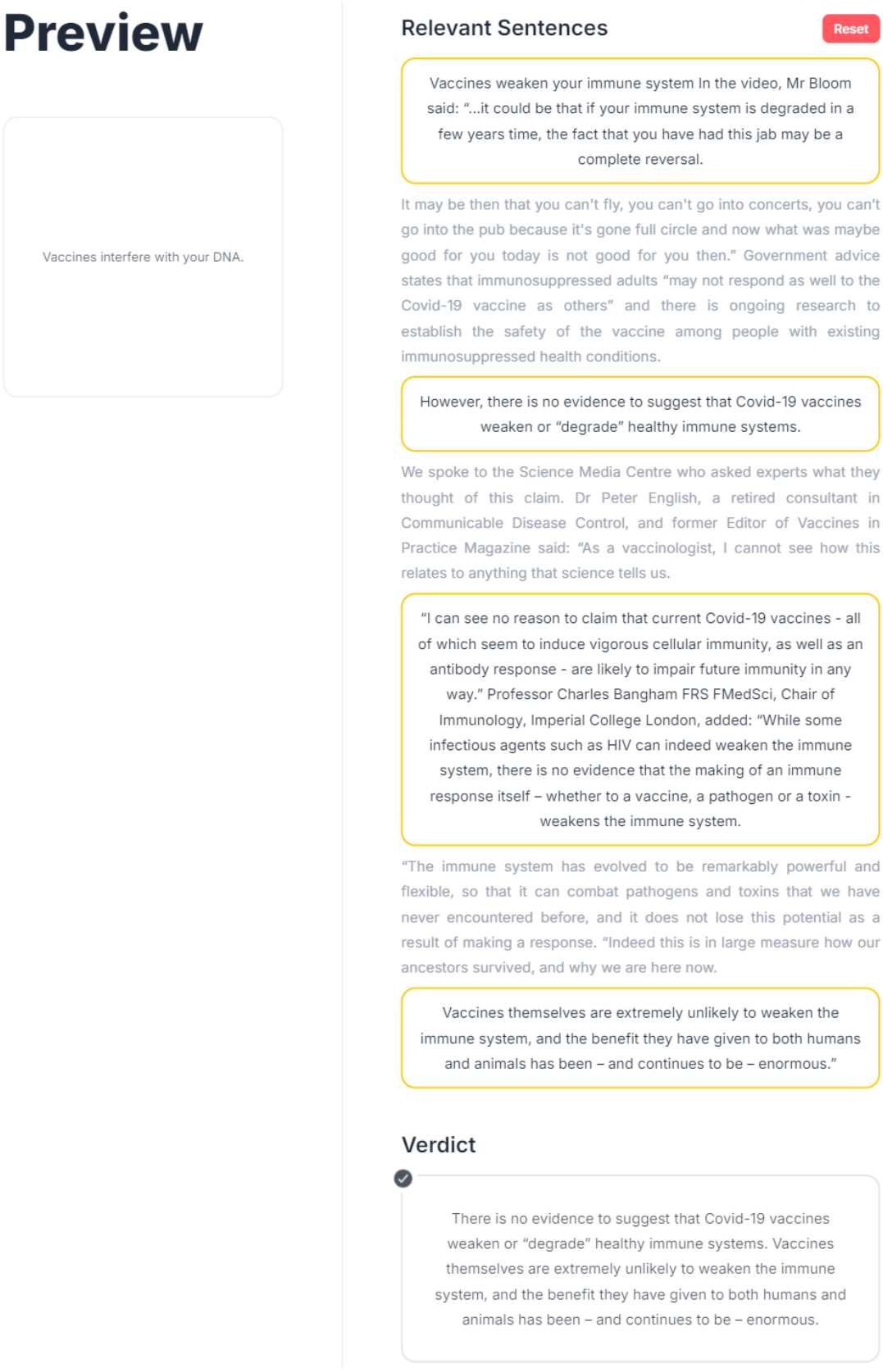

Figure 24 - Verdict generation tool results

4. Back-end

This section provides a detailed explanation of the implementation of the AI4TRUST Platform's back-end and communication framework. It covers the technologies used, the handling of requests between the front-end (web app) and WP3 services, the structure of their endpoints, and the automated generation of these endpoints.

4.1. Implementation

The back-end component serves as an interface that connects each service's REST API to the platform's standardised endpoints, facilitating seamless interaction between WP3 services and the front-end module.

It is built using Spring Boot[18], a robust Java-based framework that simplifies the development of microservices by providing production-ready features with minimal configuration. Maven[19] is used for project management. It ensures a consistent project structure, simplifies the integration of external libraries, automates the build process, and supports the scalability and maintenance of the project throughout its development.

Figure 25 provides an overview of the platform's communication flow.

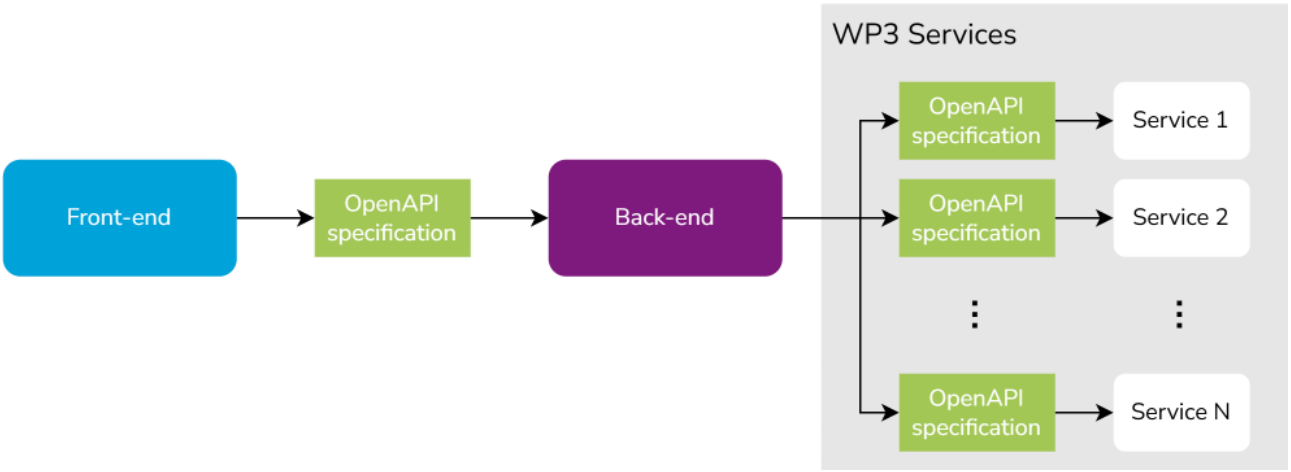

Figure 25 - API request diagram

The front-end component interacts with the back-end, which in turns adjusts and forwards the front-end requests to each WP3 service.

[18] https://spring.io/projects/spring-boot

[19] https://maven.apache.org/

Each component communicates through a set of REST API layers defined using the OpenAPI[20] specification. This approach supports API implementation across various development contexts, encourages the use of automatic code generation tools, standardising the exchange of data between the front-end and back-end components, and helps reduce SW bugs. For the AI4TRUST platform, OpenAPI generator[21] was used.

Starting from the OpenAPI definition, a server stub was generated, offering an initial skeletal implementation of the server-side functionalities. This includes the foundational structure for request handling, defining routes and endpoints, as well as establishing data structures. This approach streamlines the development process by allowing the focus to shift towards implementing core business logic, while also ensuring compliance with the defined API standards.

We also utilised OpenAPI Generator to create the client API, ensuring adherence to established patterns and facilitating proper communication between the front-end and back-end components, thereby eliminating the need for manual implementation and minimising errors.

The back-end project organises the code into three distinct layers: the model, responsible for defining the data structure, which in this case is auto-generated; the controller, which handles incoming HTTP requests and connects the API calls to the business logic; and the service, where the core business logic is implemented, processing the data and interacting with the model. This pattern promotes separation of concerns and enhances maintainability and scalability.

An illustration of the back-end project structure can be seen in Figure 26.

```
├── src/
│   ├── main/
│   │   ├── java/ai4trust/backend/
│   │   │   ├── audio/
│   │   │   │   ├── AudioConstants.java
│   │   │   │   ├── AudioController.java
│   │   │   │   └── AudioService.java
│   │   │   ├── video/
│   │   │   │   ├── VideoConstants.java
│   │   │   │   ├── VideoController.java
│   │   │   │   └── VideoService.java
│   │   │   ├── image/
│   │   │   │   ├── ImageConstants.java
│   │   │   │   ├── ImageController.java
│   │   │   │   └── ImageService.java
│   │   │   ├── text/
```

[20] https://www.openapis.org/

[21] https://openapi-generator.tech/

```
│   │   │   │   ├── TextConstants.java
│   │   │   │   ├── TextController.java
│   │   │   │   └── TextService.java
│   │   │   ├── auth/
│   │   │   │   ├── AuthConstants.java
│   │   │   │   ├── AuthController.java
│   │   │   │   └── AuthService.java
│   │   │   ├── version/
│   │   │   │   ├── VersionConstants.java
│   │   │   │   ├── VersionController.java
│   │   │   │   └── VersionService.java
│   │   │   ├── languages/
│   │   │   │   ├── AudioTranscriptionLanguage.java
│   │   │   │   ├── VerdictGenerationLanguage.java
│   │   │   │   ├── ...
│   │   │   │   └── LanguagesController.java
│   │   │   ├── security/
│   │   │   │   ├── KeycloakClientRoleConverter.java
│   │   │   │   └── SecurityConfig.java
│   │   │   ├── configurations/
│   │   │   │   ├── CheckWorthyClaimDetectionConfig.java
│   │   │   │   ├── KeycloakConfig.java
│   │   │   │   └── ...
│   │   │   ├── errors/
│   │   │   │   ├── exceptions/
│   │   │   │   │   ├── InvalidCredentialsException.java
│   │   │   │   │   ├── UnsupportedLanguageException.java
│   │   │   │   │   └── ...
│   │   │   │   ├── CustomControllerAdvisor.java
│   │   │   │   ├── ErrorResponse.java
│   │   │   │   └── ErrorType.java
│   │   │   ├── utils/ObjectUtils.java
│   │   │   └── Ai4trustBackendApplication.java
│   │   └── resources/
│   │       ├── ai4trust-backend-openapi/
│   │       ├── video-anomaly-detection-openapi/
│   │       ├── ...
│   │       └── application.yml
│   └── target/
└── ...
```

Figure 26 - Back-end project structure

The **java/ai4trust/backend/** folder houses the main project. Each sub-folder, representing a specific category of the API, contains three key files: the constants file, which stores all category-

related constants; the controller, responsible for managing the connection between API calls and services; and the service file, which handles the business logic. This structure adheres to the Model-Controller-Service (MCS) pattern and is applied to the following sub-folders:

- **`audio/`**, **`video/`**, **`image/`**, and **`text/`**: Handle the API calls related to the respective data types;
- **`auth/`**: Manages API calls for user authentication, specifically between the back-end and Keycloak;
- **`version/`**: Manages API calls related to platform version requests;
- **`languages/`**: Contains a file for each tool defining compatible languages, along with a controller for managing API calls related to the platform's language list.

The **`security/`** folder includes the `KeycloakClientRoleConverter.java` file, which is responsible for parsing Keycloak's response to enforce role-based access checks. Additionally, it contains a configuration file that implements the Spring Security filter chain, managing the platform's authentication and authorization processes. For additional information, please refer to the Spring Security Architecture[22].

The **`configurations/`** folder contains the configuration files for each tool, where the bean methods are defined to manage dependency injection and application context setup (as per Spring **`@Configuration`** annotation[23]). This structure ensures that the necessary objects are properly instantiated and managed within the Spring framework.

The **`errors/`** folder includes the definitions of known errors, an exceptions sub-folder where specific exceptions are defined, and a controller[24] dedicated to handling those exceptions. This setup ensures a consistent approach to error handling across the application.

The **`utils/`** folder contains an `ObjectUtils.java` file, which aggregates a collection of functional utilities designed for use throughout the project.

The `Ai4trustBackendApplication.java` file contains the entry-point of the application.

The **`resources/`** folder contains all the openAPI specifications for the platform and WP3 services, along with an `application.yml` file responsible for managing the Spring properties and configuration.

The **`target/`** folder contains the OpenAPI generated sources.

[22] https://docs.spring.io/spring-security/reference/servlet/architecture.html
[23] https://docs.spring.io/spring-framework/reference/core/beans/java/configuration-annotation.html
[24] https://docs.spring.io/spring-framework/reference/web/webflux/controller/ann-advice.html

4.1.1. Security

For authentication, the system employs Keycloak[25], an open-source identity and access management solution, alongside a PostgreSQL[26] database for user persistence.

The entire API authorization process is handled through Spring Security 6[27], which offers built-in support for common security concerns, and together with Keycloak, can handle user authentication, role-based access control, and protection against common vulnerabilities like cross-site request forgery[28] (CSRF).

API calls are authenticated through JWT[29] generated by Keycloak, and WP3 services are also authenticated through specific API keys saved securely in the environmental variables of the backend hosting machine.

An overview of the various layers of security of the AI4TRUST Platform is illustrated in Figure 27. After the front-end obtains the JWT through the login operation, it requests an API by sending the JWT via HTTPS to the back-end. The back-end employs Spring Security to validate the JWT with Keycloak, filters the request based on the user's role, and subsequently forwards the request to the WP3 service using the corresponding API key.

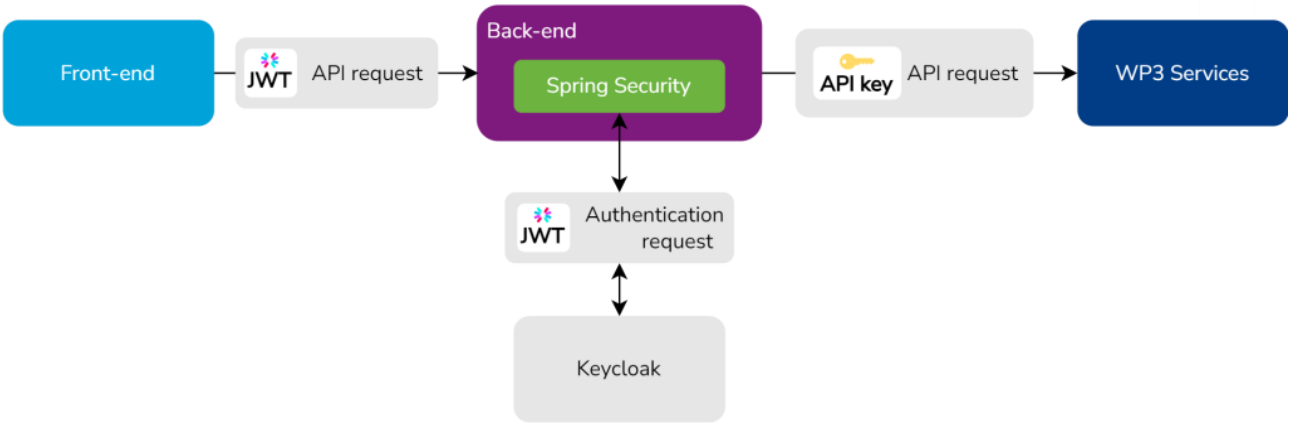

Figure 27 - API request diagram

[25] https://www.keycloak.org/
[26] https://www.postgresql.org/
[27] https://docs.spring.io/spring-boot/how-to/security.html
[28] https://developer.mozilla.org/en-US/docs/Glossary/CSRF
[29] https://jwt.io/

4.2. OpenAPI specification

The OpenAPI definition between front-end and back-end defines the main access point for the platform services, keeping systems and applications agnostic of each service's REST API specification, while ensuring security and access control.

The AI4TRUST platform end-points are listed in the next sections, and are grouped based on their purpose. Each endpoint (except for "/version" and the "/auth/login" endpoint) requires some sort of authentication. In general, the endpoints defined for the WP3 services require a bearer token (JWT) to be accessed, while most of the authentication end-points require the refresh token (HTTP-only cookie), both obtained through the login step.

4.2.1. Platform

The platform end-points are used to obtain information related to the platform such as the current version number and the platform available languages.

- GET /version
  This end-point returns the current version of the application. It is useful to check compatibility between components (i.e. front-end-to-back-end compatibility).
- GET /languages
  This end-point returns all the available languages of the platform. It represents a superset of all the languages available for each WP3 service.

4.2.2. Authentication

The authentication end-points are used for the user-based authentication of the platform. All the end-points except for the login uses Http-only cookie[30] authentication.

- POST /auth/login
  The login end-point requires a **username** and a **password**, which are used to authenticate the user and returns a JWT
- POST /auth/refresh
  The refresh end-point is used for requesting a new JWT. Useful when the current JWT expires and the user is still logged on.
- DELETE auth/logout
  The logout end-point logs out the current user.

[30] https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

- POST auth/status
  The status end-point takes as input a JWT and returns its status.

4.2.3. Image

The image end-points are used to request a service on images.

- POST /image/deepfake-detection
  The image deepfake detection endpoint takes an **image URL** as input (through a query parameter), and starts an AI-generated image detection job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**
- GET /image/deepfake-detection/status/{req_id}
  The image deepfake detection status end-point takes a **request id** as input (through path), and returns the status of the request. The status can be "PROCESSING" if the job is still running, "COMPLETED" otherwise.
- GET /image/deepfake-detection/report/{req_id}
  The image deepfake detection report end-point takes a **request id** as input (through path), and returns the AI-generated image detection results.

4.2.4. Video

The video end-points are used to request a service on video-based content.

- POST /video/deepfake-detection
  The video deepfake detection endpoint takes a **video URL** as input (through a query parameter), and starts a video deep-fake detection job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**
- GET /video/deepfake-detection/status/{req_id}
  The video deepfake detection status end-point takes a **request id** as input (through path), and returns the status of the request. The status can be "PROCESSING" if the job is still running, "COMPLETED" otherwise.
- GET /video/deepfake-detection/report/{req_id}
  The video deepfake detection report end-point takes a **request id** as input (through path), and returns the video deepfake detection results.
- POST /video/reverse-video-search
  The reverse video search endpoint takes a **video URL** as input (through a query parameter), and starts a reverse video search job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**

- GET /video/reverse-video-search/status/{req_id}
  The reverse video search status end-point takes a **request id** as input (through path), and returns the status of the request. The status can be "PROCESSING" if the job is still running, "COMPLETED" otherwise.
- GET /video/reverse-video-search/report/{req_id}
  The reverse video search report end-point takes a **request id** as input (through path), and returns the reverse video search results.

4.2.5. Text

The text end-points are used to request a service on text-based data.

- POST /text/disinformation-signals-detection
  The text disinformation signals detection end-point takes as input a **title**, a **text** paragraph and the content **language** and returns the labels ("OFFENSIVE", "SENSATIONAL", "HATE") associated with the portions of the input.
- POST /text/check-worthy-claim-detection
  The text check-worthy claim detection end-point takes as input a **text** paragraph and the content **language** and returns a label representing the check-worthiness level of the content provided.
- GET /text/verdict-generation
  The text verdict generation end-point takes as input (through query parameters) a **claim**, a fact-checking **article**, the number of relevant **sentences**, the writing **style** (journalistic or social), and the **language** of the content, and returns a text-based verdict.
- GET /text/verdict-generation/styles
  The text verdict generation styles end-point returns the available styles of the verdict generation service.

4.2.6. Audio

The audio end-points are used to request a service on audio-based content, including videos.

- POST /audio/anomaly-detection
  The audio anomaly detection endpoint takes a **video URL** as input (through a query parameter), and starts an audio anomaly detection job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**
- GET /audio/anomaly-detection/status/{req_id}
  The audio anomaly detection status end-point takes a **request id** as input (through path),

and returns the status of the request. The status can be “PROCESSING” if the job is still running, “COMPLETED” otherwise.

- GET /audio/anomaly-detection/report/{req_id}
  The audio anomaly detection report end-point takes a **request id** as input (through path), and returns the audio anomaly detection results.
- POST /audio/deepfake-detection
  The audio deepfake detection endpoint takes a **video URL** as input (through a query parameter), and starts an audio deep-fake detection job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**
- GET /audio/deepfake-detection/status/{req_id}
  The audio deepfake detection status end-point takes a **request id** as input (through path), and returns the status of the request. The status can be “PROCESSING” if the job is still running, “COMPLETED” otherwise.
- GET /audio/deepfake-detection/report/{req_id}
  The audio deepfake detection report end-point takes a **request id** as input (through path), and returns the audio deepfake detection results.
- POST /audio/transcription
  The audio transcription endpoint takes a **video URL** as input (through a query parameter), and starts an audio transcription job. To keep track of the status of the job and to retrieve the finished result, it returns a **request id.**
- GET /audio/transcription/status/{req_id}
  The audio transcription status end-point takes a **request id** as input (through path), and returns the status of the request. The status can be “PROCESSING” if the job is still running, “COMPLETED” otherwise.
- GET /audio/transcription/report/{req_id}
  The audio transcription report end-point takes a **request id** as input (through path), and returns the audio transcription.

5. Infrastructure

5.1. System

The platform is designed to accommodate both the needs of researchers, focused on data collection and analysis, and the requirements of the pilot, which is centred around the integration of AI tools.

As such, we deployed an infrastructure composed of two different and dedicated systems:

- a scalable compute and storage system, dedicated to data collection, analysis and processing;
- a dedicated, fixed size, isolated virtual machine, dedicated to the execution of the pilots.

5.1.1. Data and processing platform

The core platform for AI4TRUST is designed around containerized workloads, deployed on a scalable, hybrid environment, which collect, transform, analyse and store data in a modern data lakehouse.

Consequently, the computing layer is based on Kubernetes, the state-of-the-art solution for modern cloud architectures, hosted on Azure (North Italy region).

Thanks to the scalability and efficiency of the cloud platform, we could start the environment with a minimal configuration composed of:

- 3 server with 8 virtual cpus and 32GB Ram
- based on Ubuntu 22.04
- dynamic storage

This allows us to optimise the Total Cost of Ownership (TCO), keep the baseline costs low but at the same time quickly scale up to handle specific tasks (e.g., experiments) or spikes in the data flow. As of August 2024, the hardware resources are adequate for the workloads of the initial data collection: YouTube, Telegram, and online news.

The core platform deployment is a requirement for the whole duration of the project: an expansion plan is currently under definition, to properly plan the growth of the system and handle the increasing amount of data and processing required from the workloads.

We plan to scale both horizontally, by adding new nodes and enlarging the storage, and vertically, by growing the size of the machines, following the increasing compute needs of the platform.

5.1.2. Pilot hosting

The pilot, due to its target and configuration, is focused on the integration of AI tools, which are hosted on partners' premises and invoked via APIs.

As such, the requirements for running the system are minimal, dictated only by the needs of the frontend/backend application.

The system is thus deployed on a dedicated virtual machine, hosted in the same cloud (Azure - North Italy region), with the following configuration:

- 1 server with 2 virtual cpus and 8GB Ram;
- Operating system: Ubuntu 24.04;
- Storage: 30GB SSD.

Given that the application focuses on integrating external tools via APIs, there are no needs for local processing nor data storage. As such, there is no provision for scaling the system.

The deployment is currently planned for the duration of the first pilot, and will be closed after its ending.

5.2. Repository

The following section introduces the structure of the repositories, the formalisations and design patterns. The goal is to obtain a structured approach for repository management, environment setup, and collaboration across multiple repositories. Additionally, it covers tracking component compatibility and managing issues to ensure seamless integration and efficient collaboration among partners.

For the AI4TRUST project, we use the distributed version-control system Git together with GitHub[31].

5.2.1. Repositories organisation

GitHub facilitates the grouping of multiple repositories through organisations[32]. Within the AI4TRUST project, repositories shall follow a standardised naming convention:

- **AI4TRUST-WP[number]-[repository-name]**: it identifies the Work Package (WP) owner and the specific service;
- **AI4TRUST-WP[number]-[repository-name]-openapi**: it contains the OpenAPI specification for the corresponding service.

Each user is assigned a set of permissions for managing their respective services, along with read-only access to all repositories (including those they do not own). This structure enhances collaboration and ensures project-wide visibility and awareness across all Work Packages.

[31] https://github.com

[32] https://docs.github.com/en/organizations

5.2.2. Versioning

To simplify change tracking and ensure interoperability between components, a semantic versioning[33] system is envisaged. This versioning format uses three numerical identifiers to represent the development stage of each component, as follows:

M.m.p

- **Major (M):** Incremented when a component introduces changes that are incompatible with one or more linked components from the previous version;
- **Minor (m):** Incremented when a component introduces changes that are fully compatible with all linked components;
- **Patch (p):** Incremented when a component applies a bug fix or error correction that remains compatible with the current linked components.

This pattern helps maintain clear version control and compatibility across project components.

A compatibility matrix, included in the README file of each component (See 5.2.4), is used to track and manage the compatibility between different components within the project. It provides a clear overview of which versions of components work together seamlessly, ensuring that any updates or changes maintain interoperability across the system. This matrix is crucial for avoiding conflicts and ensuring smooth integration as the project evolves.

5.2.3. Branching

To maintain consistency across all repositories, the following workflow pattern has been defined for branches:

- **main:** this is the primary branch that contains production-ready features. While it may still contain some bugs, it is expected to be stable and runnable at all times. All major releases are tagged from this branch;
- **dev:** this branch contains the latest in-development features. It is a staging area for features that are yet to be merged into the main branch. This branch may contain bugs and there is no guarantee of stability. It serves as an integration branch for feature development and testing;
- **dev-[feature-name]:** these are temporary branches used for the development of specific features. Although they are intended to be runnable, their stability is not guaranteed. Multiple feature branches can exist concurrently. Once a feature is complete, the following procedure is envisaged:
    1. Pull the latest changes from the **dev** branch into the feature branch;

[33] https://semver.org/

2. Manually merge any conflicts and ensure the branch remains runnable;
3. After a successful merge and verification, push the feature branch into the **dev** branch;
4. Delete the feature branch after merging.

5.2.4. Readme template

Each repository shall include a README file that provides an overview of the service, detailing its scope and its connection to the AI4Trust project. Below is an example of the suggested template for this README file (Figure 28).

```
# AI4TRUST-WP[number]-[repository-name]
This repository is part of Task T[task-number] for the [AI4TRUST project](https://ai4trust.eu/), an EU H2020 research and innovation funded initiative.
This component serves as…
<!-- Brief description of the aim of this service and link to other relevant repositories... -->

## Getting Started

### Prerequisites
<!-- List sdk/drivers/etc and how to install (link to guides), used IDE and environment for the development -->

### Installation
<!-- Installation steps -->
1. Clone this repository
2. Set the following environment variables
…

### Usage
<!-- Running instructions -->
To run the application, execute:

1. Step 1
2. Step 2
…

## Additional information

### Compatibility Matrix

**Note:** Versions are defined without the patch number

| Component 1 | Component 2 | … |
```

```
|:-----------------:|:-------------------:|                    …                    |
|                 1.x                 |                  2.x                  |  …  |
|                 2.x                 |                  5.x                  |  …  |
|                  …                  |                   …                   |  …  |


###                                                                  Documentation
<!--    Link    to    documentation,    wiki    or    relevant    material...    -->

### Branches description
<!-- Description of the repository branches -->

### Known issues
<!--      List      of      known      issues      and      workarounds...      -->

##                                                                        Authors
<!--    List    of    authors    and    the    contacts    for    support...    -->

##                                                                        Licence
<!--     Licence     description     or     link     to     the     licence...     -->
```

Figure 28 - README template

5.2.5. Issues Tracking

For issue tracking, we use GitHub Projects[34], an advanced kanban board which allows us to organise, prioritise, and track tasks, bugs, and feature requests for each service. It enables each service owner to monitor their progress in real-time, ensuring a structured and transparent workflow across all project activities.

An issue can be tagged as “Bug”, “Improvement” or “Feature Request” allowing for clear distinction between different types of tasks. These tags help prioritise issues and improve the management of development efforts. Each issue is also assigned a priority level within its respective tag, helping to focus on the most urgent tasks and efficiently allocate resources.

6. Integration and Deployment

This section outlines the integration and deployment processes for each component. It details the steps for performing local integration, the deployment procedures, as well as the integration approaches, and standards followed to ensure a positive and consistent implementation.

[34] https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects

6.1. Integration

To enable a smoother integration, each platform component (front-end, back-end and authentication services) along with its dependencies, binaries, libraries, and configuration files, are bundled within a container and managed using GitHub Packages[35]. This ensures that each component functions as a standalone microservice, promoting modularity and scalability.

The front-end is built using Vite, a modern build tool that optimises the development workflow with fast build times. The built assets are served through an Nginx server[36], which is packaged within a container named **front-end**. This container ensures that the Nginx server delivers the pre-built front-end assets consistently across different environments.

Similarly, the back-end is compiled and packaged using Maven. The resulting application is executed within a container called **back-end** that runs a Java Runtime Environment.

The authentication services (Keycloak and PostgreSQL) are also managed through container images, and act as a microservice for the back-end.

6.1.1. Local integration

During the development process, the platform components are executed on a local machine. The setup is the following:

- Keycloak and PostgreSQL services are orchestrated locally using Docker Compose;
- Depending on the integration requirements, either both the front-end and back-end projects, or just one of them, may be executed locally;
- If local integration is not required for the front-end or back-end, the latest images from GitHub are used instead.

Once the development is complete, the platform components are bundled within a container, ready to be deployed (see the following section).

[35] https://github.com/features/packages

[36] https://nginx.org/en/

6.2. Deployment

The deployment server (see Section 5.1.2) is accessed via SSH, adhering to strict security standards, including private-public key authentication. It hosts containers for the front-end, back-end, Keycloak, and PostgreSQL, and the reverse proxy. The containers are deployed using a Docker Compose[37] configuration file, which instructs the Docker Engine[38] to manage and orchestrate the services.

A visualisation of the final deployment structure is shown in Figure 29:

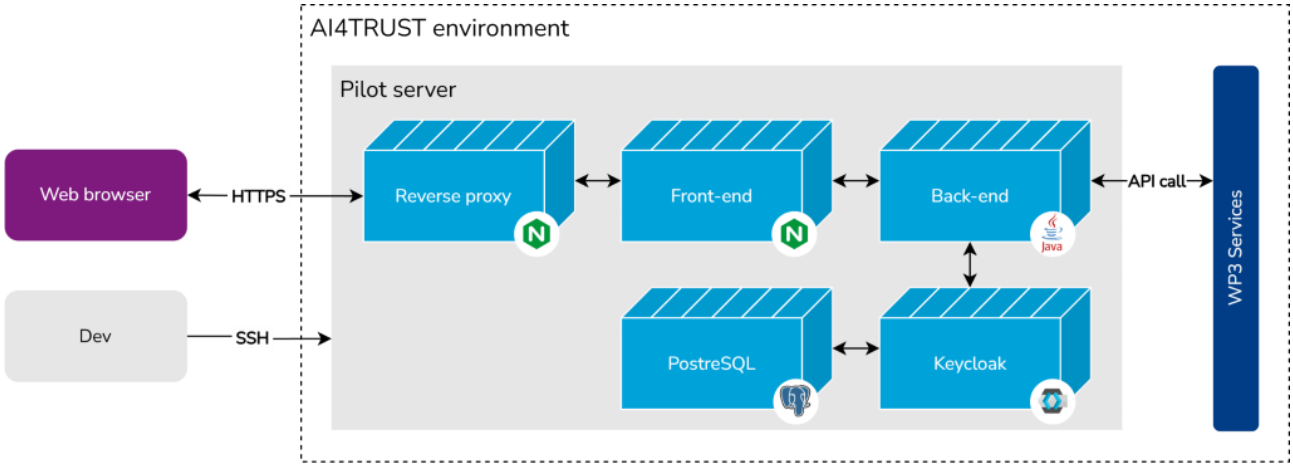

Figure 29 - Deployment structure

As shown, the front-end is accessed by users (via web browsers) through a reverse proxy. Built with Nginx, this reverse proxy enables secure HTTPS connections using signed SSL certificates, ensuring encrypted communication between clients and the server and addressing common issues such as CORS-related problems.

The authentication flow is managed by the Keycloak and PostgreSQL containers, which are accessed by the back-end. Each WP3 service is accessed via secure API calls, ensuring safe and efficient communication. For development purposes, the server can also be accessed via SSH.

[37] https://docs.docker.com/compose/

[38] https://docs.docker.com/engine/

7. Conclusions and Recommendations

The first version of the AI4TRUST platform is in place and operational and this deliverable provides a detailed analysis of its parts. In particular, this document describes the frontend, analysing the functionalities offered to the end-user and its implementation, and the backend, with a specific focus on its implementation, the integration with the frontend and the partners' services, and the security measures put in place. Moreover, this document provides insights about the chosen infrastructure and the adopted deployment process.

As a result, this report shows that the first version of the AI4TRUST platform is ready for the forthcoming pilot evaluation that assesses the value of the employed "AI-driven data analysis methods". This enables the collection of end-users' feedback, allowing to improve the above mentioned methods that will be employed in the foreseen automated pipeline characterising the next releases of the platform (that will be described in the forthcoming "D5.6 - AI4TRUST Platform v2" and "D5.7 - Final AI4TRUST Platform").