Funded by the Horizon 2020 Framework Programme of the European Union
SocialTruth - Grant Agreement 825477

D2.3 Refined Distributed System Architecture

Dissemination Level:	**PU**
Nature of the Deliverable:	**R**
Date:	**05/03/2020**
Distribution:	**Internal**
Editors:	**UTP**
Contributors:	**UTP, Thales, ESF, QWANT, Z&P, TECOMS, ICCS, LSBU**
Dissemination Level:	**CO**

**** Dissemination Level:*** *PU= Public, RE= Restricted to a group specified by the Consortium, PP= Restricted to other program participants (including the Commission services), CO= Confidential, only for members of the Consortium (including the Commission services)*

***** Nature of the Deliverable:*** *P= Prototype, R= Report, S= Specification, T= Tool, O= Other*

Disclaimer

This document contains material which is copyright of certain SocialTruth consortium parties. All SocialTruth consortium parties have agreed to the full publication of this document.

Neither the SocialTruth consortium as a whole, nor any certain party of the SocialTruth consortium warrants that the information contained in this document is capable of use, or that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using the information.

The contents of this document are the sole responsibility of the SocialTruth consortium and can in no way be taken to reflect the views of the European Commission. The European Commission is not responsible for any use that may be made of the information it contains.

The commercial use of any information contained in this document requires a license from the proprietor of that information. For information and permission requests, contact the SocialTruth project coordinator Dr. Konstantinos Demestichas (ICCS) at cdemest@cn.ntua.gr.

The content of this document may be freely distributed, reproduced or copied as content in the public domain, for non-commercial purposes, at the following conditions:

it is requested that in any subsequent use of this work the SocialTruth project is given appropriate acknowledgement with the following suggested citation:

"Deliverable 2.3 Refined Distributed System Architecture (2020)" produced under the SocialTruth project, which has received funding from the European Union's Horizon2020 Programme for research and innovation under grant agreement No.724087. Available at: http://www.socialtruth.eu"

a) *this document may contain material, information, text, and/or images created and/or prepared by individuals or institutions external to the Socialtruth consortium, that may be protected by copyright. These sources are mentioned in the "References" section, in captions and in footnotes. Users must seek permission from the copyright owner(s) to use this material.*

Revision History

Date	Rev.	Description	Partner
01/01/2020	1.0	Initial Version	UTP
10/02/2020	1.1	Contribution by Z&P	Z&P
17/02/2020	1.2	Contribution by Qwant	Qwant
18/02/2020	1.3	Contribution by ESF	ESF
19/02/2020	1.4	Contribution by THALES	THALES
20/02/2020	1.5	Document refinement and integration of partners contribution	UTP
21/02/2020	1.6	Document refinement after internal review	UTP
23/02/2020	1.7	Additions by TECOMS. New subsections by UTP.	UTP
25/02/2020	1.8	Additions by ICCS, document finalization	UTP, ICCS
27/02/2020	1.9	Pre-final version	UTP
13/03/2020	1.10	Review and alignment with D2.1 based on 1st review recommendations	UTP
23/03/2020	1.11	Adaptations based on partners feedback	UTP
27/03/2020	1.12	Peer review and Pre-final version v.2	UTP
02/04/2020	1.13	Final version	UTP

List of Authors

Partner	Author
UTP	Michał Choraś, Rafał Kozik, Marek Pawlicki, Krzysztof Samp, Paweł Ksieniewicz, Michał Woźniak, Rafał Renk
TECOMS	Guido Villa
Z&P	Giulia Venturi, Alessandro Zanasi, Davide Mauro Ferrario
ESF	Nahid Oulmi
QWANT	Stan Assier
THALES	Romain Ferrari
LSBU	Chathura Galkandage
ICCS	George Koutalieris

Table of Contents

Index of figures

Glossary

AMQP	Advanced Message Queuing Protocol
API	Application programming interface
CQRS	Command Query Responsibility Segregation
CRUD	Create Read Update Delete
CSV	Comma-separated values
EMVE(s)	Expert Meta-Verification Engine(s)
GDPR	General Data Protection Regulation
GPS	Global Positioning System
GPU	Graphics Processing Unit
HMI	Human-Machine Interface
HTTP(s)	Hypertext Transfer Protocol (Secure)
IP	Internet Protocol
J2SE	Java 2 Platform, Standard Edition
JDK	Java Development Kit
JSON	JavaScript Object Notation
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
P2P	Peer-to-Peer
RAG	Red Green Amber
RF	Random Forest
RNN	Recurrent Neural Network
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WP	Work Package
XML	Extensible Markup Language

Executive Summary

This document (D2.3) exists to provide detailed, functional and non-functional specifications of the distributed SocialTruth Architecture and the related interfaces, taking into account the use-cases and scenarios of T2.1. The document also tackles the technical, security and social & human aspects of SocialTruth. It is intended for the end-user partners, who will deploy the platform, the project researchers and developers, who will be providing the technical solutions, and to the platform integrators. D2.3. produces the outcomes to deliverables D3.4-5, D4.1-3, and D5.2.1-3. The motivation, scope, relation to other deliverables and the intended audience is fully explained in Section 1. In the subsequent sections the document addresses general approach to architecture design, including such aspects as information flow, integration of verification microservices, description of particular key building blocks constituting the SocialTrurth platform. The document focuses also on the security, privacy, socio-technical and human aspects that are significant from the architectural viewpoint.

It is also worth mentioning that the current document is the follow-up of M6 initial version of the SocialTruth architecture delivered at the early stage of the project and that detailed specification of particular modules will be provided in respective deliverables of technical WPs (WP3-WP5).

This deliverable has been developed also taking into consideration the revised version of D2.1 that was produced during March 2020 in order to address the recommendations of the experts' review process related to the specifications of the trust and blockchain aspects. In D2.3 we address the software design aspects or blockchain related components. Implementation details are provided in WP4.

1 Introduction

This document combines the outcomes from the following tasks:

- Task 2.2: Distributed Architecture Design – led by UTP,
- Task 2.3: Security & Privacy by Design – led by TECOMS, and
- Task 2.4: Socio-technical and Human Aspects – led by Z&P.

It is the refined version of the SocialTruth distributed architecture.

1.1 Motivation

The motivation of this document is to present the general architecture of the SocialTruth solution, allowing the consortium (and community) to further work on implementation and prototypes.

SocialTruth is a distributed platform to evaluate the credibility of the content (inserted for analysis by the user) allowing for fake news detection.

The task of SocialTruth is to give some hints that the content is not credible; not to decide which is true and which is not (especially since in the post-truth era, each single fact might have various interpretations and descriptions).

1.2 Intended audience

This deliverable is a report produced for all the members of the SocialTruth project. Specifically, the results of this report are addressed to the following audience:

- End-user partners, who will deploy elements of the SocialTruth platform and its particular components,
- The SocialTruth project researchers and developers, who will provide technical solutions,
- The platform integrators.

1.3 Scope

In general, the purpose of the D2.3 document reporting this task is to provide detailed, functional and non-functional specifications of the distributed SocialTruth Architecture and the associated interfaces with the outputs of Task 2.1 (scenarios specification, use-cases and requirements) taken into account.

In details, D2.3 is divided into three parts, which will focus on:

- Technological aspects: specification of the SocialTruth components and interfaces with the focus on modularity, flexibility and openness (microservice-oriented approach is considered),
- Security aspects: addressing security and privacy needs in technical specification, including such techniques as data encryption, privacy keys, digital signatures, security of the communication (protocols), authentication, anonymization, etc.
- Social and human aspects affecting system design, in particular the design and optimization of HMIs for SocialTruth addressing the needs of different user categories.

1.4 Relation to other deliverables

D2.3 is the final iteration of the SocialTruth architecture, with a description of key building blocks of the SocialTruth platform, its design and functionalities.

This deliverable is linked with other deliverables produced within the SocialTruth project. The relations within WP2 outputs and tasks have been shown in Figure 1.

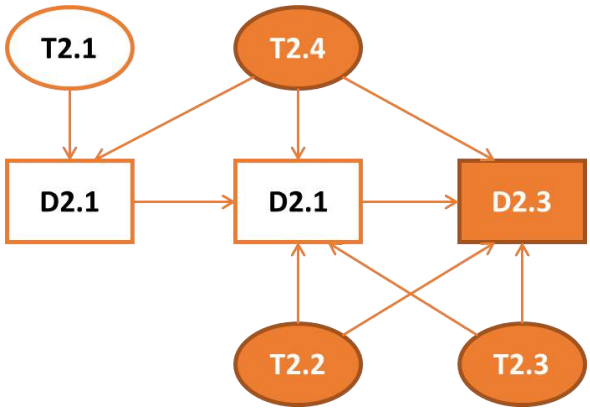

Figure 1 - Relation of D2.3 to other deliverables and project tasks.

The D2.3 deliverable produces the outcomes to the following deliverables:

- D3.4/D3.5 – SocialTruth Content Analysis and Verification Services – Release 1 and 2,
- D4.1 SocialTruth Blockchain
- D4.2 SocialTruth Lifelong Learning Expert System
- D4.3 SocialTruth Digital Companion
- D5.2.1/5.2.2/5.2.3 SocialTruth Integrated Prototype R1.0/2.0/3.0

2 SocialTruth Refined Platform Architecture

2.1 General logical overview

This section provides some general specification of the key components that compose the SocialTruth platform. As the starting point for the SocialTruth platform architecture design specification we have used the general model depicted in the Description of Action (see Figure 2).

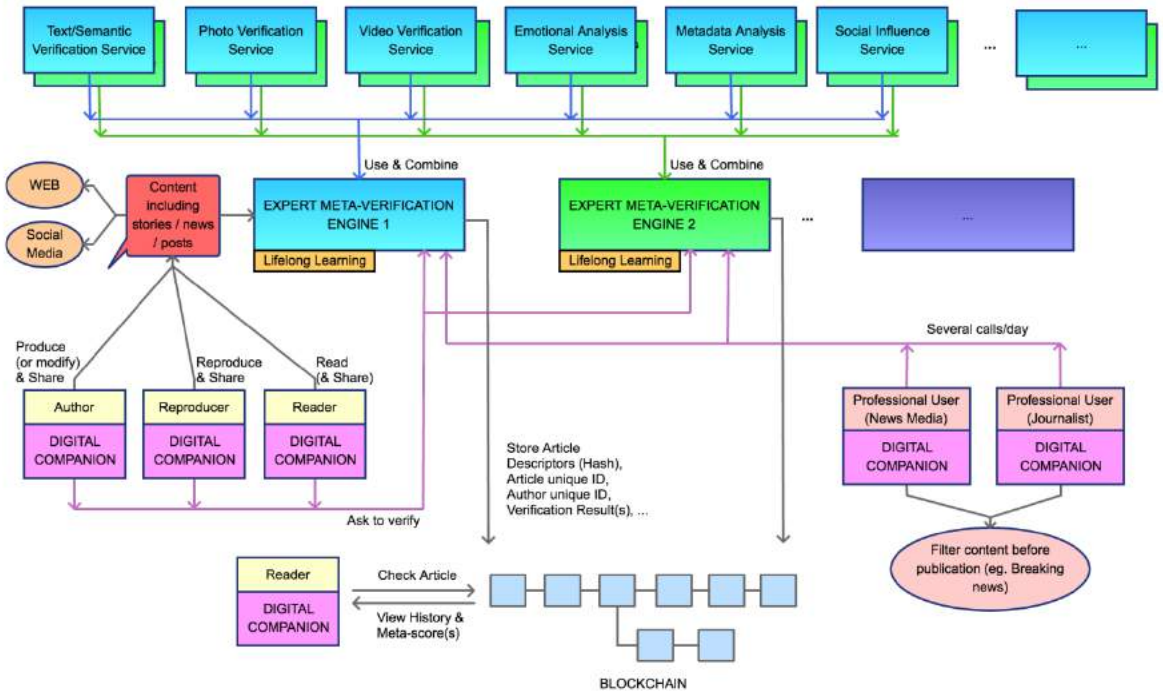

Figure 2 - SocialTruth Platform Architecture [source: DoA]

2.1.1 Key components

The components within the platform are:

- **Digital Companion**: considered as a browser plugin that allows a non-professional user to invoke a meta-verification process upon some form of digital content (e.g. an article), passing its URI as an input to a meta-verification engine. In case of the non-professional use, the Digital Companion can be used by the author of the digital article, by a reproducer (who shares the article in the Social Media) or even by a simple reader of the article, who wishes to get an estimation of the credibility of the content before or after reading it. In case of the professional use, the Digital Companion allows several calls per day to the APIs of the meta-verification engine(s). SocialTruth will follow a user-centred design approach to product for the Digital Companion.
- **Verification Services**: a set of heterogeneous distributed verification services providing a specific type of content analytics (e.g. for text, image, video) or verification-relevant functionality (e.g. emotional descriptors, social influence mapping). Some of these services are made available and deployed by the SocialTruth consortium partners, while others are coming (either in open source or not) from third-party service providers. Each service can be deployed at a different hosting

facility (e.g. different servers or clouds), hence there is no imposed centralization. All of them use the same standard SocialTruth interfaces to allow them to be easily accessible, reusable and interchangeable. The registry of service providers and the services they offer is stored and maintained in the blockchain.

- **Expert Meta-Verification Engine(s) - EMVE:** to combine verification results from various verification services to compute a meta-score that reflects the credibility of the digital content under consideration. It follows an open design, open algorithms and an expert-systems approach. It uses open algorithms while most of its settings and weights (e.g. which verification services to prefer or to avoid, with what priority, etc.) can optionally be configured through its standard web-service interfaces.
- **The SocialTruth blockchain**: a distributed system of records with respect to the digital content verification history. Since the complex web and social media landscape is characterised by several competing content creators and distributors, each with their own motives, interests, strategies and practices, the blockchain is an ideal tool to establish reputation and trust without the need of a central authority or intermediary (thus also avoiding centralizing even more regulatory power to the US Internet giants, such as Facebook or Google). Hence, a public distributed ledger provides an auditable and immutable trail of verification actions and reputation scores. The blockchain will store article identification information, article descriptors (e.g. hash codes for digital content integrity), author identification information, verification and meta-verification scores, as well as identification information for the verification services that have been used to calculate them. It will also hold the registrar of verification service providers and the services they offer.

Functionally, these elements depend on one another and are logically pile-up as classical N-tier architectural model that is comprised of a data layer, a business layer, and a presentation layer. The bottom data layer of the SocialTruth N-tier architecture model constitutes Verification Services together with common interfaces providing access to the data. The middle layer, providing business logic is the Expert Meta-Verification Engine (EMVE). Finally, the presentation layer capabilities are provided by the Digital Companion component. Each of these functional components is further composed of dedicated modules that provide or facilitate the dedicated functionalities the specific component is intended to provide.

2.1.2 Microservice architecture style

The growth in container solutions has resulted in the development of microservice solutions for deploying dedicated applications which can be integrated in the main platform using standards protocols. One of the critical characteristics of microservice deployments is the reliance on carrying out unit testing on individual components without any external dependencies. This feature has also contributed to the creation of distributed and flexible workflow architecture that partly enables/disables component instantiation without affecting the overall performance of the platform. The distribution of micro-services packaged in containers has also resulted in achieving scalability as several instances of a single component can be deployed in run-time.

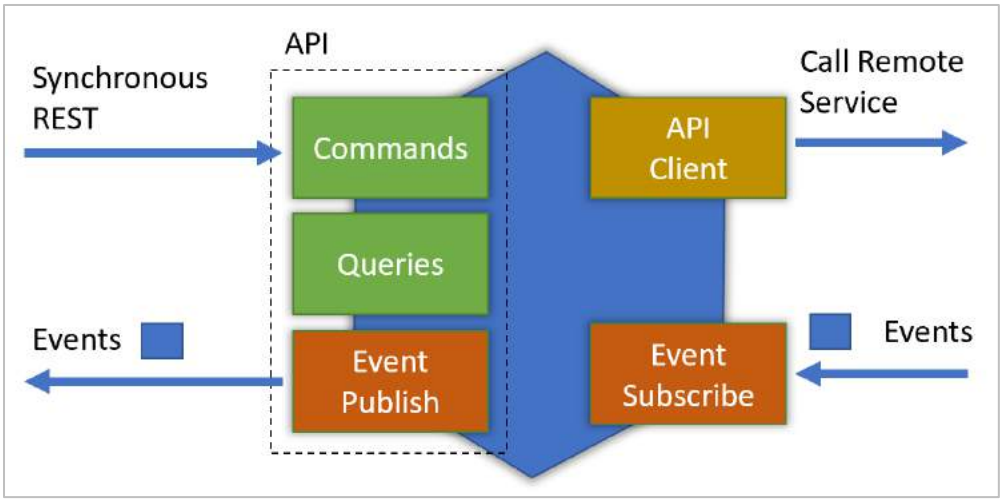

Figure 3 - Microservice – the atomic elements composing SocialTruth platform

As it is shown in Figure 3, the key elements building the microservice components are:

- API that allows other clients to interact with the service in synchronous (e.g. REST) and asynchronous ways (e.g. events).
- Client API for interacting with other components/services.
- Event subscriber (listener) that allows the service to retrieve notification from other services.
- Internal and private storage that maintains all relevant data required for serving the purpose of the microservice.

The synchronous API calls can be essentially divided into commands and queries types. The command is a type of a remote invocation that internally mutate the data of the service. For instance, service may expose methods which add, update or delete some data. Another type of API calls constitutes queries, that can be essentially used to find specific data using various search criteria. Such calls do not mutate internal data.

Microservices adapt the single responsibility paradigm and promote loosely coupling. There are different ways as to how the monolithic application can be divided into several smaller autonomous components. The most obvious strategy is to use decomposition that is based on business capability. For example, a system supporting sales would be decomposed into services responsible for customer, orders, invoices, etc. In general, the process of decomposition produces smaller entities that can be developed individually by separate teams. This allows the teams to sustain autonomy in terms of architectural patterns and technologies selected to develop a specific service.

Microservice-based approach is a concept that is gaining in popularity. However it must be noted here that various pros and cons exist. Analysing various microservices based solutions, several architectural challenges can be identified:

- Decomposition related to the problems of breaking application into smaller, autonomous and independent pieces.

- Integration related to the problems of communicating the services with each other and presenting the scattered data to the consumers.
- Database-related issues of database architecture in the microservices environment (in particular distributed transactions).
- Monitoring and observability of the distributed systems.

2.2 Blockchain-enabled distributed environment

The SocialTruth platform is built upon a distributed architecture allowing for optimal information propagation. Blockchain technology has been chosen to fulfil this requirement.

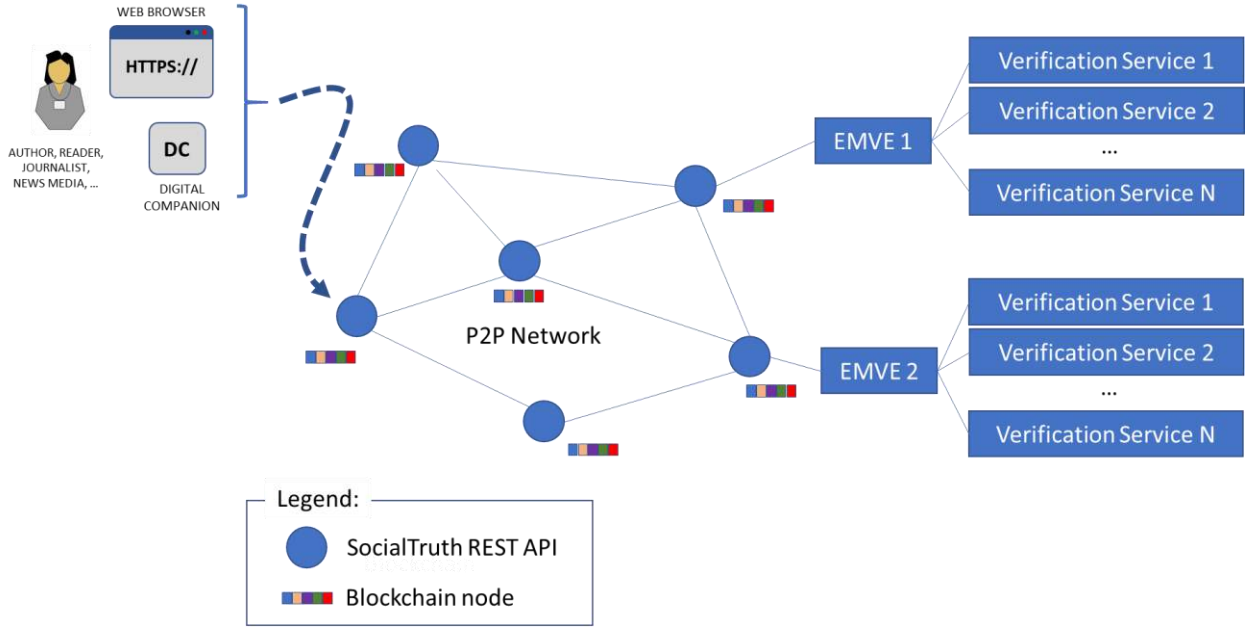

Figure 4 - Blockchain Integration

The Figure 4 depicts how Blockchain technology is integrated into the global SocialTruth architecture. In order to have support for decentralisation, the SocialTruth architecture foresees that each Blockchain node will integrate an API that will handle the calls from the SocialTruth client (i.e. the digital companion) and call the appropriate components.

This Blockchain node component is part of the SocialTruth P2P network. It acts as a distributed ledger and interacts with other Blockchain nodes within the network in order to synchronize all the information about the fakeness scoring of each item computed through the SocialTruth service. Each entity that wants to join the SocialTruth network will have the possibility to run their own Blockchain node alongside the gateway.

This blockchain mode will be responsible for the information sharing within SocialTruth. The data stored into the Blockchain will be defined by (but is not limited to):

- Url: the item url that has been checked (in other words: the checked content)
- Hash: the hash representing the content of the item (article, video, picture, etc…)
- Score: the fakeness/credibility score
- The verification services used to verify the content
- The settings chosen/used by the user
- Last_check: the time that this item was last verified

In order to interact with the blockchain node component (read or write entries), one must go through the gateway component.

The gateway component is the entry point to the SocialTruth P2P network and thus the gateway to the SocialTruth decentralized data. Each gateway is part of the Blockchain Node. It offers a JSON REST API to access the SocialTruth Blockchain. The endpoints of this API are:

- */url* - This is typically called by the digital companion
    - POST method to request the fakeness score of an entry

```
{

“url”: “http://www.socialtruth.eu/#how_works”,

“hash”:
“aa540c3c3c6a928e60d14bd7c51e2338646454a9da3989491ad291c2af96b9db5d91373f62
1bd43800bf2ac37e2ce61be2e582cebb28ca0b74780773871db4e8”

}
```

    -
- */entry* – This is called by the MetaVerificationSystem after the computation of the fakeness scoring
    - POST method to insert a new entry

```
{

“url”: “http://www.socialtruth.eu/#how_works”,

“hash”:
“aa540c3c3c6a928e60d14bd7c51e2338646454a9da3989491ad291c2af96b9db5d91373f621
bd43800bf2ac37e2ce61be2e582cebb28ca0b74780773871db4e8”,

“score”: 0

}
```

    - PUT method to ask for a new check of an existing entry

```
{

“url”: “http://www.socialtruth.eu/#how_works”,

“hash”:
“aa540c3c3c6a928e60d14bd7c51e2338646454a9da3989491ad291c2af96b9db5d91373f62
1bd43800bf2ac37e2ce61be2e582cebb28ca0b74780773871db4e8”,
```

```
"score": 0
}
```

2.3 SocialTruth workflow (information flow)

A typical workflow will be as follows:

1. A user enters/opens digital companion.
2. In the Digital Companion users' interface, a link allows the users to access the EMVEs users' interface via which the user can:
   - choose preferred EMVE (Expert Meta Verification Engine) from the list of available EMVEs – at least one EMVE should be available. Each EMVE has a set of verification services to be used.
   - choose/disable some verification services, and change their settings (see Fig. 6)
3. User provides an URL to the digital companion to be checked by EMVE (and its verification services).
4. The digital companion computes a hash of the content from the URL, and sends the data (alongside with the calculated hashes) to EMVE.
5. The EMVE calculates/computes answer (credibility score) using chosen verification services and provides the result to user.
6. EMVE stores the result (the link, hash, settings, used verification services, results) in blockchain

The above steps are illustrated in the following diagram (Figure 5).

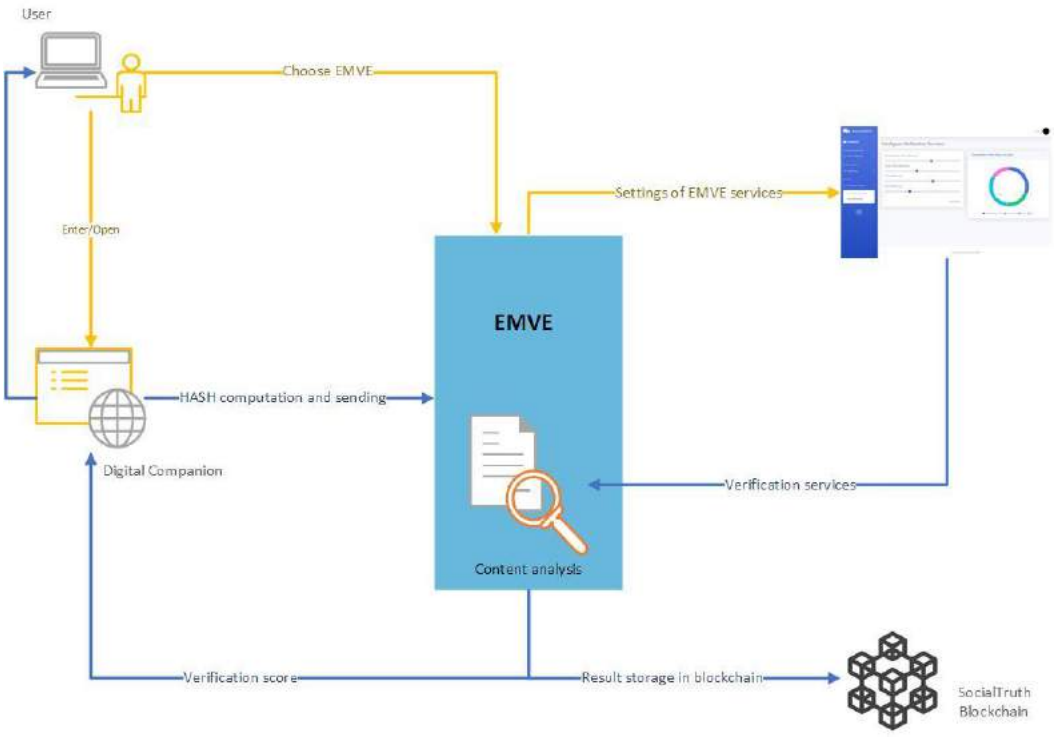

Figure 5 - SocialTruth workflow diagram

The mock-up version (early prototype) of the Step 4 is presented in Figure 6 .

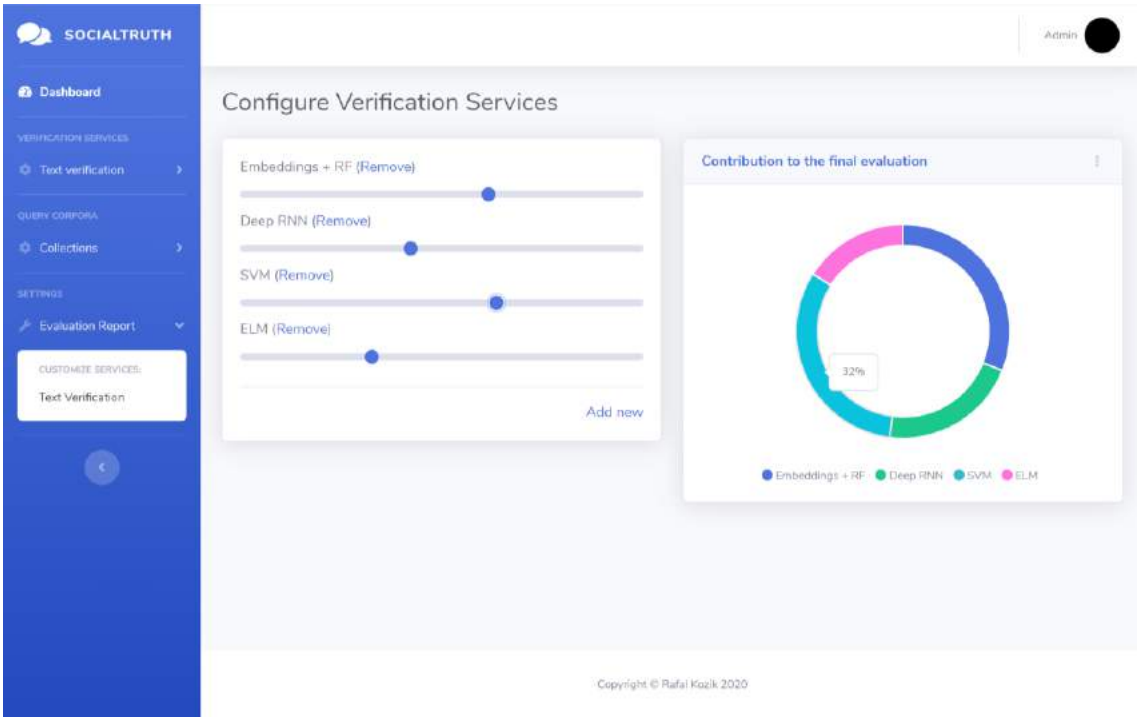

Figure 6 - The mock-up of choosing verification service and changing their settings

The following also applies:

1. In between points 5 and 6, EMVE checks if the exact content was earlier verified by the same services and settings. If so, it returns the result without computations. If the verified services and/or settings are changed, the computations are being done, and the new entry will be stored in blockchain.
2. User can check the same content many times using different EMVE(s).
3. Different EMVE(s) do not communicate with each other.

2.3.1 Architectural and technological view

From a broader perspective, it is important to explain the environment where the proposed solutions will operate and how they will bring benefits for the end-users (e.g. press agencies or web portals), that are all kind of actors that need to cope with fake news challenges. Therefore, in this section, we give a general overview of the distributed platform for fake news detection, which has been depicted in Figure 7.

The technology stack has been decomposed into the following logical elements that have been detailed in the next subsections:

- physical elements (nodes) and their orchestration,
- verification services,
- messaging and event processing.

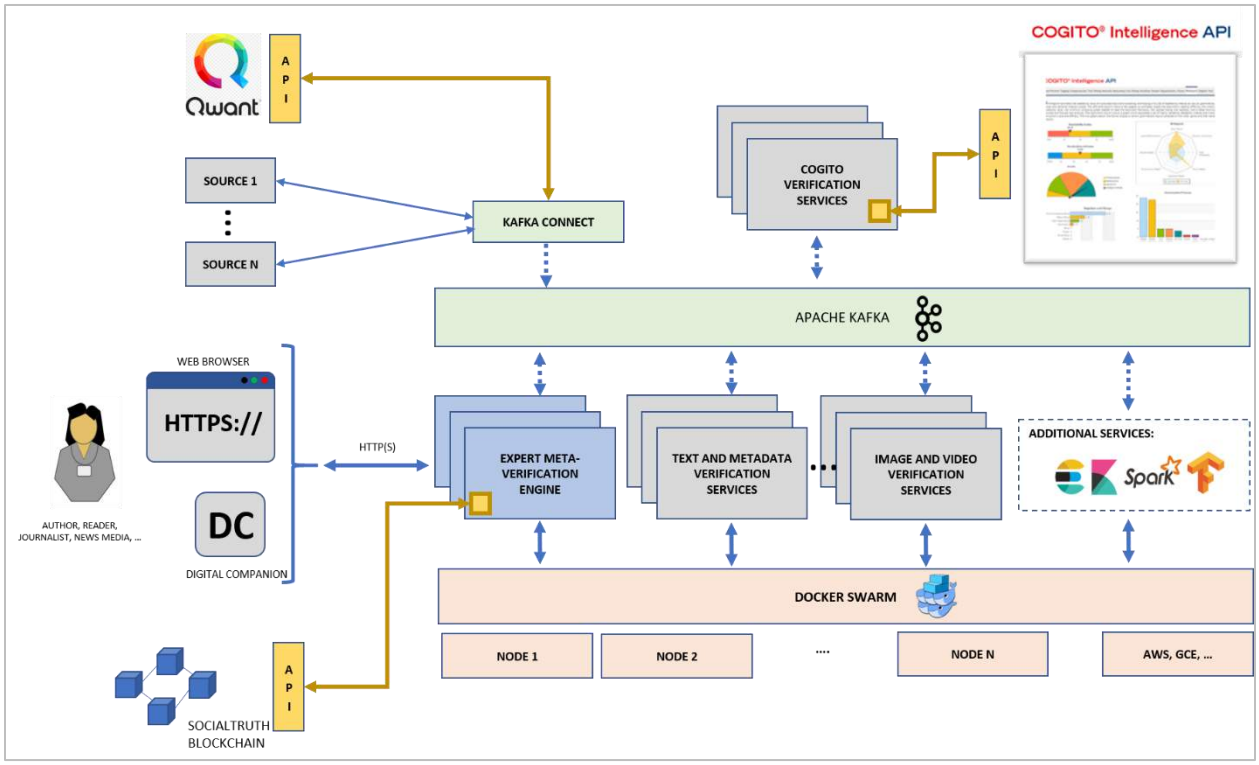

Figure 7 - The SocialTruth Platform – technology stack overview.

There are the following assumptions and observations coming form the diagram shown above::

1. The data is ingested into the platform either by the user (journalist, author, reader, etc.) over HTTP(S) protocol or by using dedicated crawlers (data connectors) that send data over the binary protocol to the Apache Kafka framework.
2. The Apache Kafka is a distributed streaming platform implementing the publish-subscribe model.
3. Once the ingested data is published to one of the Kafka topics, it can be simultaneously consumed by various verification services and/or stream processing applications.
4. Once the services finalize their computations, they make the results available on another Kafka topic, which can be consumed by other services again.
5. Physically, EMVEs do not have to be deployed at the same location together with Verification Services These can subscribe to remote Kafka brokers using secure communication channels and interact with Verification Services.

There are additional elements that in details are related the EMVE, Digital Companion, and SocialTruth Blockchain. The details are given in the sub-sections 2.6-2.8.

2.4 Physical nodes comprising the system

The first and the most bottom layer in the technology stack constitutes the orchestration framework. It is laid down on top of an infrastructure composed of virtual and hardware machines. This layer is intended to implement automated resource management and thus it facilitates the entire platform with such capabilities as flexibility, scalability, and fault tolerance. It is the responsibility of the orchestration layer

to effectively deploy the services on the available computational nodes (both physical and virtual). It is achieved thanks to the containers that are sandboxes that contain the implemented service, together with all the software dependencies (libraries and execution environment). In such a form, the services can be easily migrated between the computational nodes and deployed. In the proposed solution, we have used Docker Swarm system.

In order to address the scalability and the platform orchestration, we recommend using Docker together with Docker Swarm to maintain the ecosystem. The concept of using Docker Swarm in the SocialTruth architecture is presented in Figure 8.

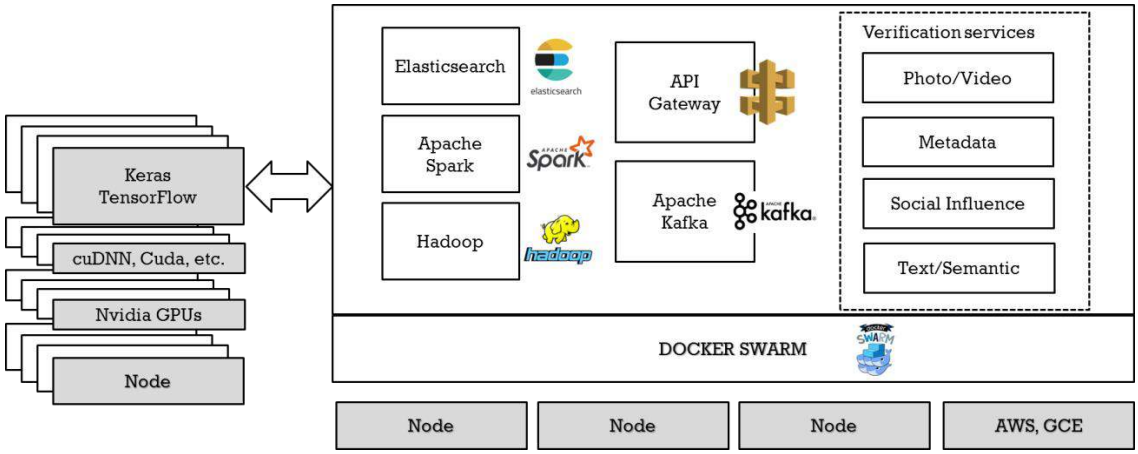

Figure 8 - Use of Docker Swarm in the SocialTruth architecture

The Docker Community is a forum for enthusiasts that use the virtual containers, micro services and distributed applications. Moby is an open framework created by Docker to assemble specialized container systems. It provides a "lego set" of dozens of standard components and a framework for assembling them into custom platforms. Docker is an open source project that is aimed at simplifying the deployment of an allocation by means of containers. It allows for building an image with the application deployed inside. A running instance of image is called a container. The image contains all the dependencies that the applications need to run (e.g. operating system, runtime environment, specific system libraries, etc.). The image can easily be run anywhere (on the variety of host operating systems) executing the application in an isolated environment. Swarm is a Docker-native container orchestrator used to manage Docker containers as a cluster of machines. Docker Swarm eases the deployment, organization, management and scaling of Docker containers.

The containerisation differs from hardware virtualization in the way that it has higher performance (containers do not emulate the entire computer architecture), lower resource consumption, and smaller images (containers do not require a full operating system). Containers solve many problems of software delivery, such as runtime environment configuration, isolation, application management, and portability. Using a single image one can run many containers (copies of the same application). At the same time Docker enables rapid "diff" changes within the various software builds to verify the consistency of the solution over the versions. An image is a stateless building block of the Docker system. From the functional point of view, an image has a layered structure. It means that images are easy to extend by adding

additional layers. For example, a J2SE application would have a basic image of the operating system (e.g. Ubuntu). A JDK can easily be installed on top of it. Optionally, the user can also add additional programs such as Git. All modifications can be persisted afterwards as a new image. In case of Docker, it provides a dedicated configuration language that allows for quick image definition.

2.5 Integration of microservices with Apache Kafka and API Gateways

When the application is broken down into a set of separate services, eventually it so happens that these need to communicate in order to provide complex business capabilities. That capability usually needs to assemble the results obtained from multiple services. Many challenges may appear depending on the application. For example, some services need to be orchestrated to produce the final result. It means that a specific chain of actions needs to happen and these need to be sequenced in a time manner. There are two approaches that have been described in the consecutive subsections. These two approaches have their intrinsic advantages and are complementary to each other. In the proposed architecture, we anticipate a mix of both when implementing various services.

2.5.1 Orchestration

The orchestration pattern introduces a central entity (orchestrator) controlling the execution of each stage in the pipeline. In general, the orchestrator holds the code/script which indicates when and how specific services should be called and how the responses need to be aggregated. The aggregation could be performed in different ways and one of the most popular and widely used is the API Gateway. The pattern appears in many microservice frameworks such as Java Spring Cloud [1]. In general, the gateway can be seen as a reverse proxy, which is used by services that reside in the backend (are hidden behind the reverse proxy). It takes requests from the client and forwards these requests to one of the backend services. There are several advantages of using the API Gateway pattern. Firstly, it constitutes a single entry point for any call. This, for instance, allows for implementing the authorisation functionalities at the gateway. Secondly, the gateway can translate the request protocol to something else such as AMQP (Advanced Message Queuing Protocol). Thirdly, the gateway can proxy request from client to multiple services and aggregate results.

Another benefit of adapting this pattern is the fact that we can offload the microservices authentication burden directly onto a gateway. Moreover, we can abstract the microservice details (e.g. IP address) making the gateway work as a reverse proxy, mapping the user request into a specific backend service call.

This will also be beneficial from the EMVE perspective, since it will decrease coupling – the EMVEs will not have to know the location of each service. Instead, the EMVEs will use a single entry point. This will also encourage development of a consistent API.

2.5.2 Choreography

As we mentioned before, in the described system, we have adapted Apache Kafka. It is a distributed streaming platform, which enables both real-time event processing and event-driven communication between various components. From the architectural point of view, Apache Kafka constitutes a flexible

[1] https://spring.io/projects/spring-cloud

and efficient way to integrate all the components, both the existing tools as well as the new ones developed during the project or by the community.

Once the ingested data is published to one of the Kafka topics, it can be simultaneously consumed by various verification services and/or stream processing applications. Once the services finalize their computations, they make the results available on another Kafka topic, which can be consumed by other services again.

2.6 Digital Companion

2.6.1 Digital Companion User Workflow

Additionally to users' interfaces specifications in deliverable 2.2, the following user's workflow has been defined for the usage of the first version of the Digital Companion.

• Step 1: An individual user installs the Digital Companion plugin in their web browser(s). The solution will be compatible with Firefox and Chromium based browsers.

• Step 2: The individual user requests the verification of information (news/post) as described in workflow in Section 2.3 (and shown in Figure 9).

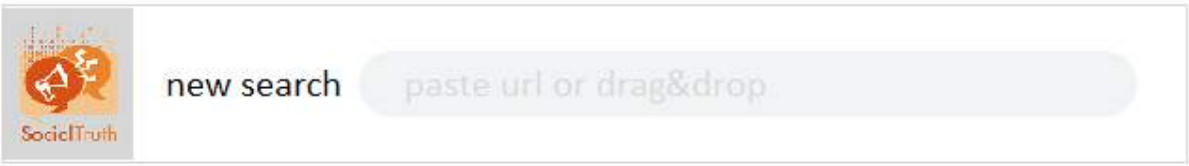

Figure 9 - Digital Companion searching interface

Immediately after giving input to the system and pressing "Enter" or the search icon, he or she should receive feedback from the system that the operation was successful (Figure 10)

Figure 10 - Digital Companion searching interface – feedback from the system

While the system collects data and information on the content, a pop up should show with a spinning wheel or a progress bar to keep the user in the loop during the loading phase.

• Step 3: The Digital Companion returns the result of verification. The result might be presented as shown by the mock-up hereafter (Figure 11).

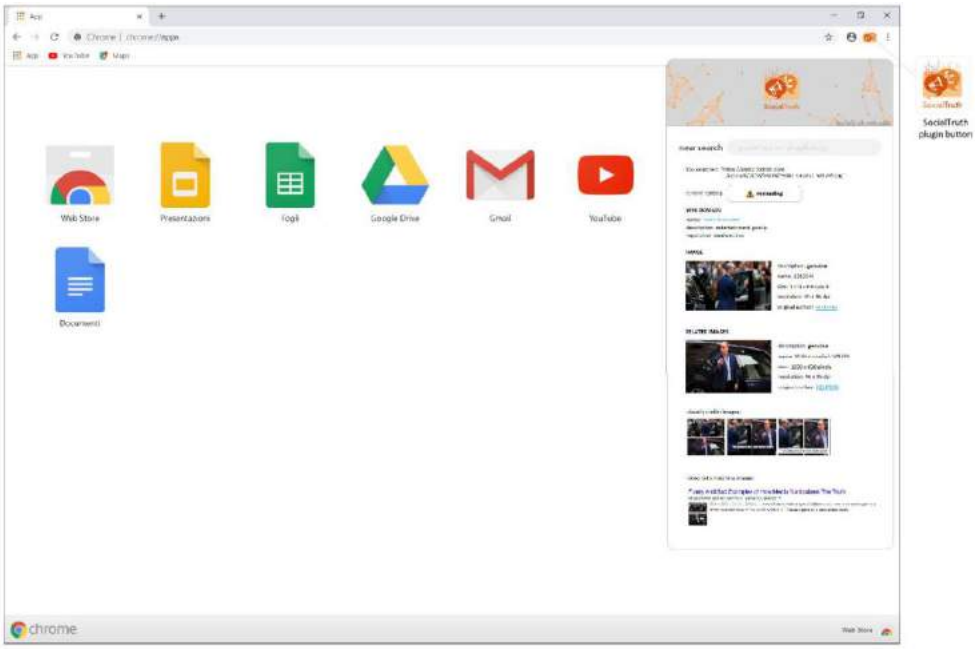

Figure 11 - Digital Companion results of verification (mock-up)

Complementary steps will be added:

• Step 4: The individual user makes their own verification. To be able to assess the accuracy of the Digital Companion instrument, the individual user will use the classic methods to verify the information: search engines, related social media posts, search for the original information; in par.

• Step 5: The individual user gives feedback on automatic verification. Based on their personal investigation, the user will appraise using a score scale the reliability of the information generated by Digital Companion, for each investigated news/post.

From the technical point of view the Digital Companion will be developed as a plugin for the latest versions of Firefox and Chromium. As an end user provides an URL to the digital companion, the digital companion computes a hash of the item content (hash function to be decided), makes a request to the EVME API at "/url" with the URL and the HASH in the JSON payload. It then waits up to 1 minute for the results (a rank from 0 to 4; 4 being fake news) from the EMVE and displays it to the end user. It has the following formats:

- API Rest (asynchronous operation with a maximum delay of 1 minute)
  - Request: POST Subject JSON integrating URL and HASH
  - Back: Return Code or JSON Object incorporating the note (5 levels) of the URL and optional Date and number of times requested
- No authentication

- 3 hard addresses in the plugin pointing to 3 different EMVE (as to address the distributed architecture).

2.6.2 Digital Companion User Preferences

In addition to the user interface functionalities, users will have the ability to configure their preferences regarding the verification services provided by Digital Companion. In the following mock-up screens this concept is illustrated. It should be noted that configurability of the provided verification services and the corresponding EMVEs is supported by the SocialTruth architecture as is explained in Section 2.3.

2.6.2.1 Digital Companion Account Settings and User Preferences

Following a standardized way to configure account settings, the user will be introduced with the following functionalities.

2.6.2.2 Digital Companion Verification Services Settings

According to Section 2.3, SocialTruth supports multiple verification services. The user can choose preferred EMVE from the list of available EMVEs and then is able to change the settings of the available verification services. This concept is depicted in the following mock-up screen (Figure 12).

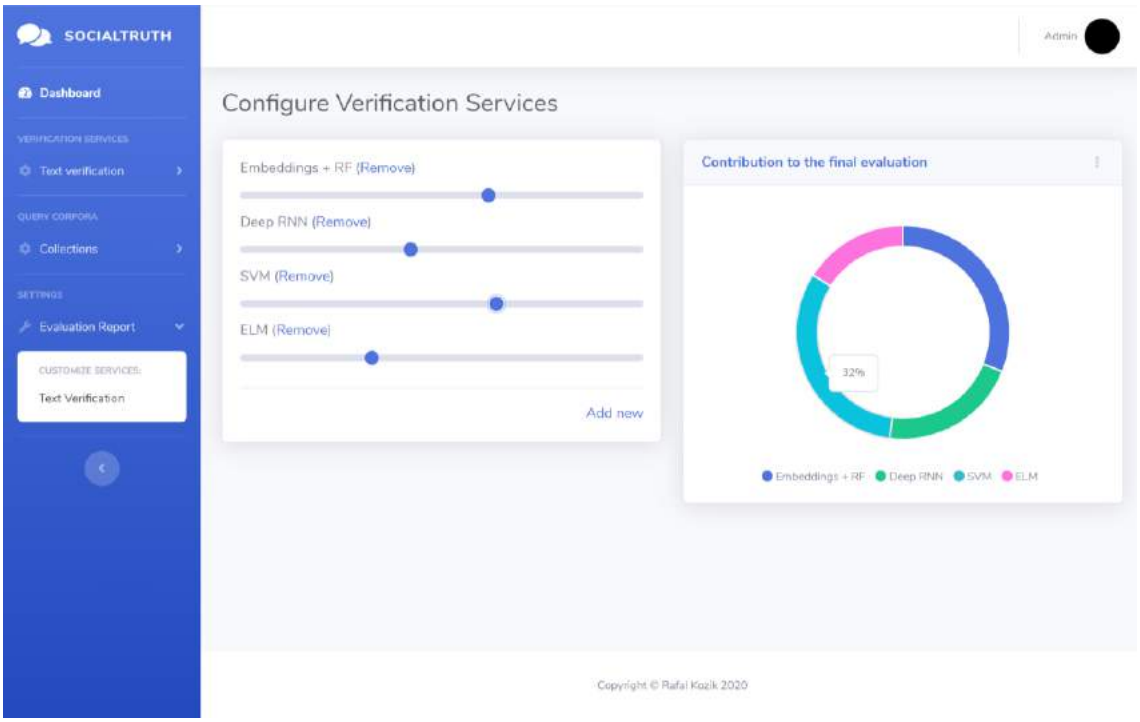

Figure 12 - Mock-up of verification preferences and user options

2.7 Verification services

2.7.1 General aspects of verification services

Verification services as such are the key building blocks of the system and are deployed as micro-services.

Micro-service is an independently deployable component, which (in the proposed architecture) is packed as a Docker container. Each micro-service is focused on providing single functionality.

Moreover, each functional service provides an API that allows other clients to interact with the service in synchronous (e.g. REST) and asynchronous ways (e.g., events). Each service can also have client API for interacting with other components/services (e.g. databases). Moreover, the service can subscribe (listen) to a notification sent from other components in the system. In the proposed architecture, we heavily use asynchronous event-based communication in favour of synchronous calls. This allows us to avoid tight coupling between the verification services and other components in the platform. In that regard, each verification service subscribes to a dedicated topic and produces results on another one.

The distributed verification system will be composed of several heterogeneous services that will be functionally focused on a specific kind of context analysis, e.g. images, text, etc. (Figure 13). From the end-user point of view these specific services should be visible as a monolithic system providing various capabilities.

In that sense, an API Gateway pattern could be used. It would allow for hiding the microservices behind the middle-layer that would be acting as a reverse proxy, handing the client requests, passing them to the specific services, and returning the received results to the client.

Another benefit of adapting this pattern is the fact that we can offload the microservices authentication burden directly onto a gateway. Moreover, we can abstract the microservice details (e.g. IP address) making the gateway work as a reverse proxy, mapping the user request into a specific backend service call. This will also be beneficial from the EMVE perspective, since it will decrease coupling – the EMVEs will not have to know the location of each service. Instead, the EMVEs will use a single entry point. This will also encourage development of a consistent API.

The services behind the gateway will be stateless entities that take the data in a predefined format and return the result. The interaction between them will also be limited. Nonetheless, the services will need several elements to facilitate their work. These include data storage, search engine, data processing, publish-subscribe systems, etc.

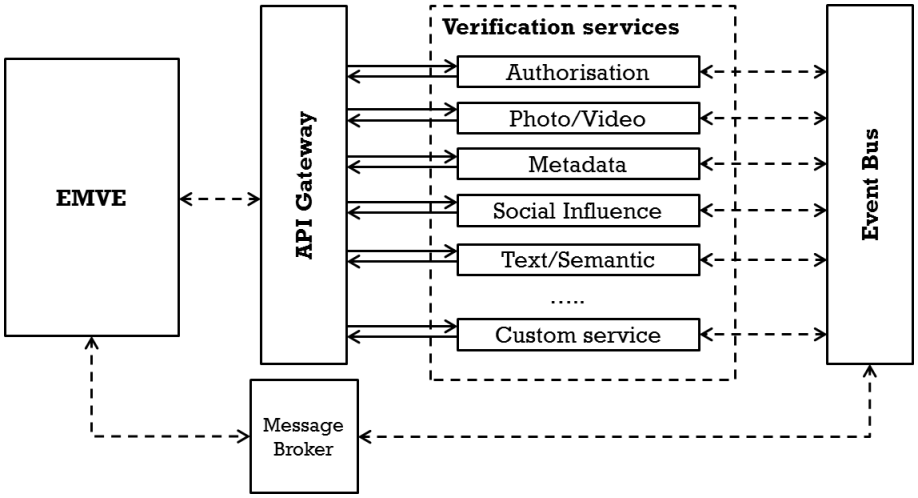

Figure 13 - SocialTruth technical architecture - distributed verification services view

Once the data is uploaded, the services can process it. Usually, it will take some time. Therefore, instead of periodically pooling the services for current state, an event-based publish-subscribe system would be a better fit in such scenarios. For example, a client requesting the verification of a specific image can submit the request via standard API and subscribe to the event bus for updates. Once the service finishes the processing it sends notification event via the event bus. Using such platforms as Apache Kafka it is possible to guarantee fault tolerance and scalability of such mechanisms. For instance, if the network connection fails or the client is down when a notification event is sent, the client is always capable of receiving the event once it is back to normal state.

2.7.2 Text Verification Services

2.7.2.1 ESF approach

The textual verification component consists in three services: style, sentiment and similarity (Figure 14).

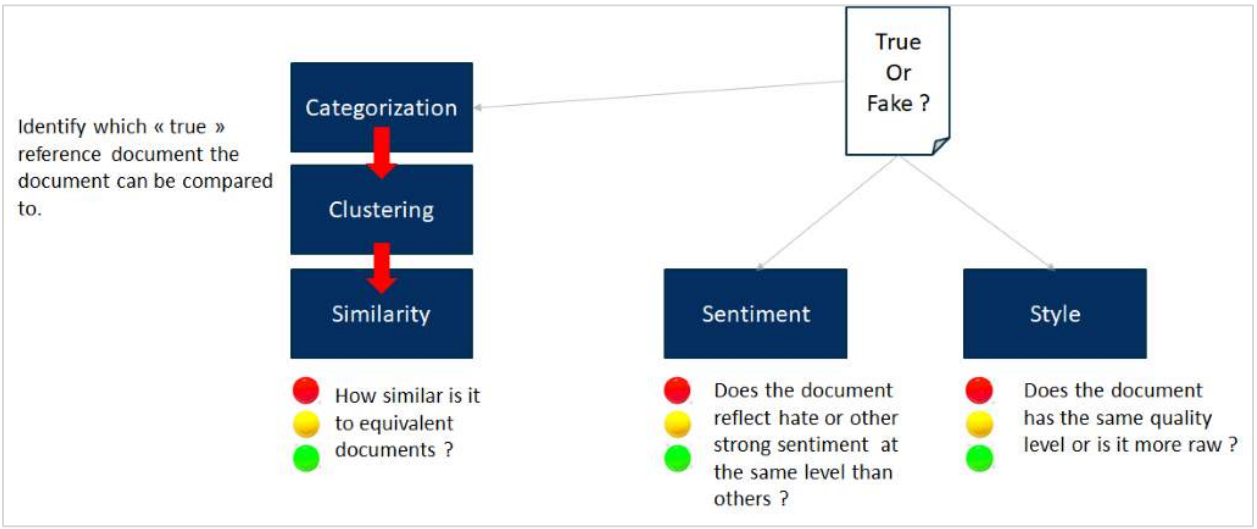

Figure 14 - ESF approach to text verificication in SocialTruth

Each of these services has the goal of providing the features discriminating fake news against true news, such as :

- Does the document provide extreme point of views/emotions ?
- Is it written in a bad style ?
- How similar are the facts presented in other trustworthy documents ?

These services will be accessible via a REST API that can be hosted on any device. The implementation and use cases of these services will be detailed in deliverable D3.4: "SocialTruth Content Analysis and Verification Services – Release 1" (M16)

Each of these services outputs will be JSON and XML files describing the document's features related to these aspects. The formal structure of these outputs will also be defined in deliverable D3.4: "SocialTruth Content Analysis and Verification Services – Release 1".

2.7.2.2 UTP approach

For the text verification scenario we plan to combine various NLP-based (Natural Language Processing) detection models that are put into a pipeline depicted in Figure 15.

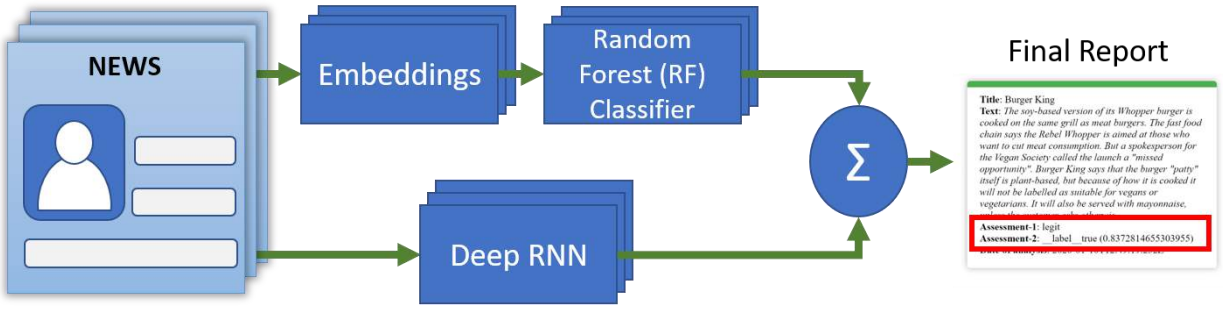

Figure 15 - The processing pipeline used for the text analysis

So far, we have proposed two fake news detectors that use entirely different machine learning approaches. The processing happens in parallel and the results from both classifiers are combined into a single report for the user.

The first used detector is based on the Deep Recurrent Neural Network (Deep RNN). To tackle the challenge of fake news, for one of the verification services a deep RNN (Recurrent Neural Network) built on top of Flair framework was used. This solution offers outstanding features in terms of neural network design, includes many state-of-the-art methods, among them numerous methods based on deep learning, also enabling GPU-based training. Flair is a Natural Language Processing library designed for all word embeddings as well as arbitrary combinations of embeddings. The crucial elements of creating the fake news detection model were carried out with the support of the Flair library. The training process was carried out based on deep learning methods after word embeddings had been carried out using the

modern and effective procedures in this area. In our work, we chose to use various types of neural networks to solve the problem of text-based fake news detection.

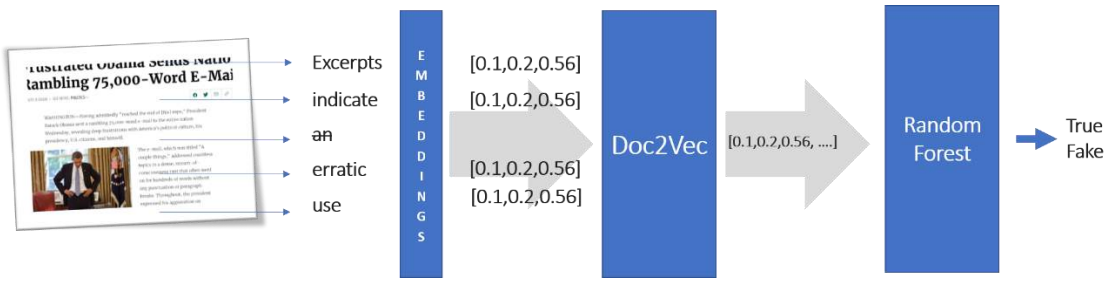

Figure 16 - Detector based on Random Forest classifier – general overview of the method.

The second used detector is based on Random Forest (RF) and its overview is presented in Figure 16. This approach is a different method from an architectural point of view. It is complementary to the Deep Recurrent Neural Network described in the previous section. The main difference is that, in contrast to RNN, the RF can be trained significantly faster or even in an online manner. It means that the model can be updated right away when a new data sample is available. This substantially increases the flexibility and makes it easier to update the entire detection model when new data is available.

2.7.3 Image Verification Services

The image verification comes in three different standard services plus an integrated services to get the results of these three standard services.

These services are:

- Copy-move detection: Do we have fake duplications of the image content subject to post processing?
- Cut-paste detection: Do we have content of two or more images in one subject to post processing?
- Erase-fill detection: Do we have anything missing in the image?

These services will be accessible via a single model file (this will be the pre-trained model) which is required to design the API to execute the model on sufficiently resourceful computing device. Making these models continually update their parameters require their API to collect more data. The implementation of the copy-move model is available in detail in deliverable D3.3: "SocialTruth Deep Learning Multimedia Verification".

On top of these services, another classifier model is designed to integrate these models by diverting a given image to different basic services and provide the final verification results. The API to this service can be directly integrated into the main architecture, as the whole image verification service will be in one API with the classifier.

2.8 Expert Meta-verification Engine (EMVE)

This services essentially combine verification results from various verification services to compute a meta-score that reflects the credibility of the digital content under consideration.

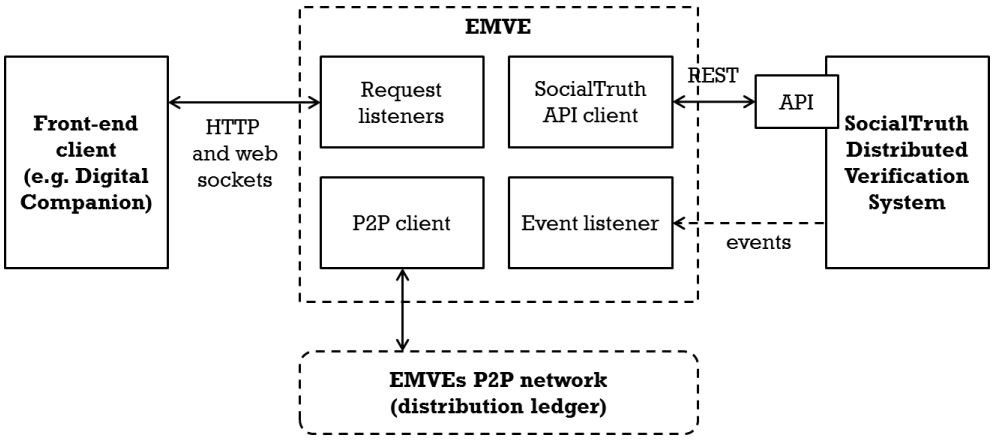

Figure 17 - SocialTruth technical architecture - single EMVE view

The Expert Meta Verification Engine (EMVE) has the responsibility to collect, gather, reconcile, organize, structure & analyse the features provided by the different SocialTruth verification & analysis services.

The input to the EMVE shall be the different responses of the SocialTruth services. These services inputs should have a structured format (JSON, XML, CSV…) that will be defined in Deliverable 4.2.

The features presented in these inputs can be:

- Continuous (any decimal variable in a range), for example: the sentiment analysis can range between [-100 : +100]
- Categorical (categorical variables that range in a discrete range). For example, an image can be Fake or Not Fake. These are two categorical variables.

In other words, EMVE uses available verification services (customized according to user settings and needs) to provide an answer of the analysed content. The mock-up early prototype can be shown in Figure 18.

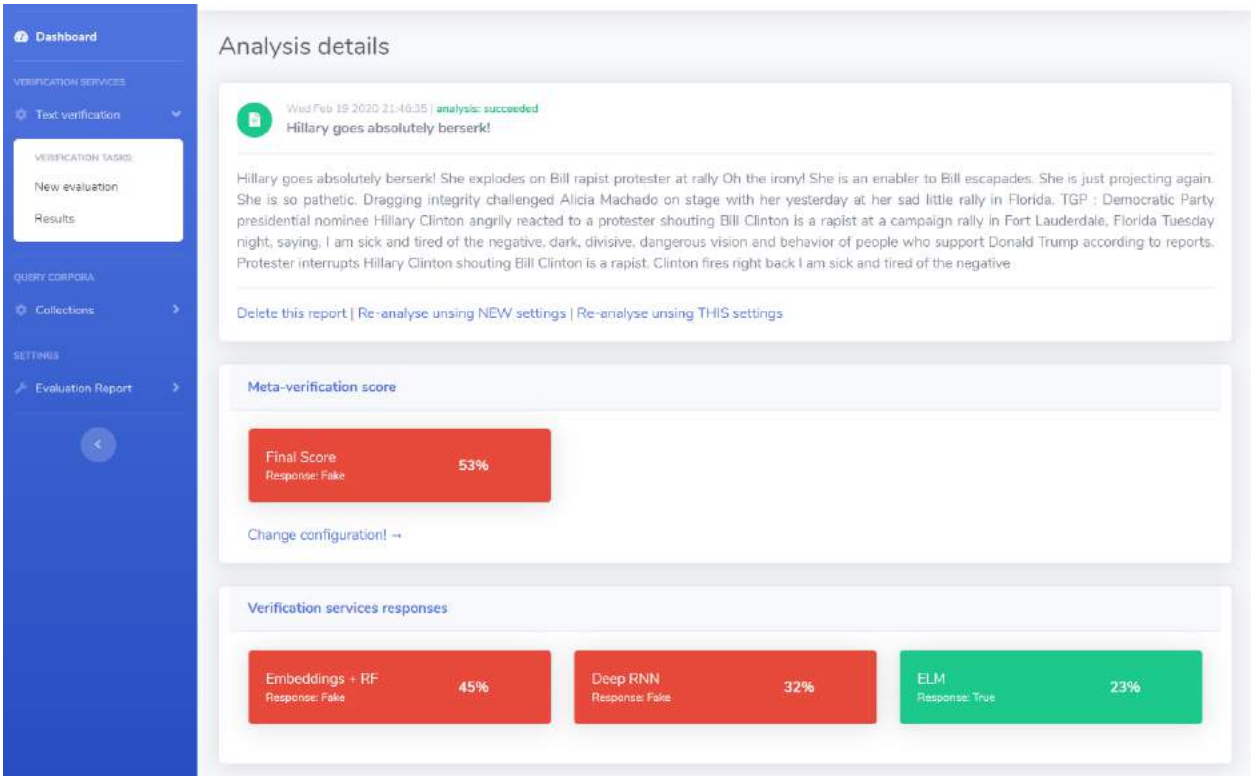

Figure 18 - Analysis details – mock-up example.

The EMVEs will need to implement the CQRS pattern (Command Query Responsibility Segregation) as the operation of querying and indexing of the document will take considerably varied amounts of time. Querying (or reading) the information of an already indexed document will be significantly faster, in particular if the document verification will be based on the URL address (e.g. we may already know that the news published on a specific website is fake and the results of analyses are already there in the SocialTruth database). In that case, the results could be returned in a simple request-response manner via the RESTful API. On the other hand, the process of indexation (document ingestion) will require a different approach. The uploaded documents need to be stored and analysed asynchronously by the services. In that case, an orchestration coordinated by the EMVE will be required. For example, it will need to start the analyses by pointing the services to the data to be analysed, wait for the results, consolidate them into a single information piece that will be consumed by the requestor. In the following scenario, a publish-subscribe messaging system would be a better fit than the request-response approach.

The EMVEs will also be the elements which are closely interacting with the end user. There are two main cases where the user will be engaged. First, the operation when user queries the EMVE to verify a particular document, news, post, etc.; second, the situation where the user wants to add a new document with the information about its credibility. Whenever the document has already been indexed and annotated by someone else, the user may also express her or his opinion regarding it.

2.9 Data models

The microservices need to store data. Depending on the context, this can also be a challenging task to implement. Defining the appropriate database architecture is problematic due to following aspects:

- Sharing a single database between microservices impacts the system scalability and services autonomy
- Some business capabilities need data that is owned by several microservices
- Implementing transaction mechanism in a distributed environment is problematic and requires coordination and extensive communication of participating microservices

The good practice says that a single database per service should be used. This means that the specific service has its own database that is isolated and is not shared with others directly. This avoids the situations where the development of one service and its data model influences the development of another service. However, it so happens that the isolated service eventually needs to reach out for the data maintained by another service. One of the options solving this would be a CQRS pattern (Command Query Responsibility Segregation. It promotes splitting the command and query parts, so that typical CRUD (Create Read Update Delete) command-based operations are handled by one system while data querying capabilities are served by the other. In order to provide query results that join data from multiple services materialised views are used. The views are updated whenever any part of the data changes. These are communicated using event busses (e.g. Apache Kafka). The service maintaining the materialised view listens to the event bus for notifications and updates the view accordingly.

There will be a certain amount of data that will need to be processed, analysed, stored, and indexed. Rather than sending the entire documents back and forth for verification, the data should be uploaded once and later referenced using pointers (e.g. URL address). Therefore, adequate data storage is an important element of the architecture.

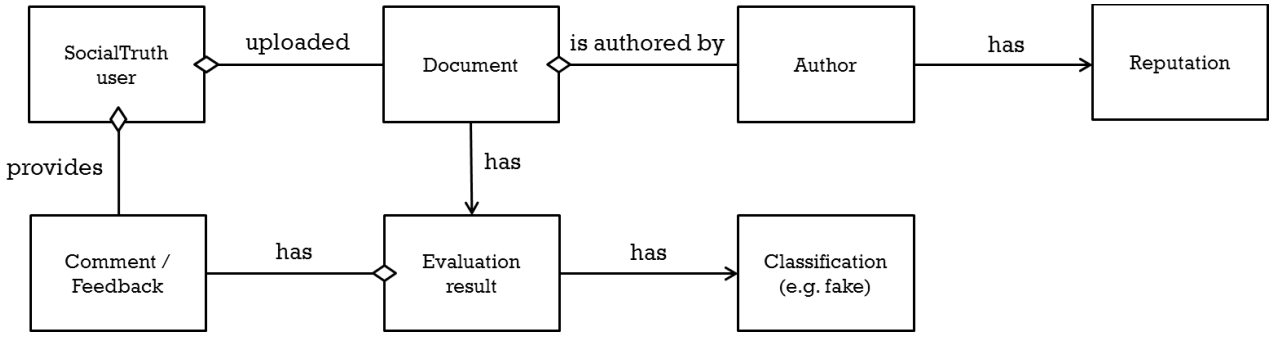

Figure 19 – The concept of data storage in SocialTruth

From the overall description of the SocialTruth proposal emerges a general data model (Figure 19) to be shared among various modules comprising the platform. The first data entity is the document to be verified, the second one is the author of this document. The third one is the reputation (verification) score. Finally, the evaluation report also constitutes an important element.

The system should also provide capabilities to look the previous verifications up. These should be indexed in various ways in order to avoid double checking of the same document. The most straightforward way would be indexing the results by source (e.g. URL address).

2.10 Monitoring and observability

In a microservice architecture a single request often spans multiple services that are hosted on separate physical servers. Each service generates a log file that is stored locally. In such a case, reconstructing the original information flow (from a client request to the returned result) could be time consuming if done manually. This is also an important aspect from the system auditing or user accounting point of view. If one needs to trace the request end-to-end, a dedicated centralized service aggregating logs is needed.

An effective approach increasing traceability would be to assign each request with a unique identifier. In case of HTTP protocols, additional header parameter could be added. Then, the identifier passed to the services can be used by them to annotate each operation stored in a log file. The log files can be efficiently shipped for central analysis and inspection using such frameworks as Elastic Stack, together with Apache Kafka. It consists of four elements:

- Beats – a sensor retrieving and capturing the operational data (the logfiles)
- Logstash – an entity ingesting the data into the Elasticsearch engine
- Elasticsearch – a data storage and indexation engine
- Kibana – a visualisation frontend for advanced analysis of the ingested data

Figure 20 - Elastic stack.

3 Security and privacy aspects

Security by design and privacy by design guidelines have been outlined in D2.2, and served as a reference in the detailed design of the various components of the SocialTruth platform, as previously described in this document.

The redefined distributed system architecture detailed in this deliverable complies with the guidelines provided in D2.2 and D7.1 regarding data security and protection: all the components of the platform have been defined taking into consideration the best data protection practices, “security by design” and “privacy by design” principles.

In detail, in compliance with the principles of data protection regulation and best practices, the following measures will be taken:

- **Lawfulness, fairness and transparency:**
  - the digital companion user will be provided with adequate notice that will detail which data will be processed and for what purpose, and any other GDPR required information. Before being able to install and use the digital companion, the user must express their acceptance by clicking on the appropriate button;
  - only for evidentiary purposes regarding the provision of consent to the processing of data, the platform will keep the following data in a dedicated log: timestamp, user IP address, port, browser used;

- **Purpose limitation:**
  - except for the evidentiary purposes referred to in the previous point, or when providing consent to the processing of data provided during the activation of the platform, no other user data that is not essential for the correct functioning of the platform will be processed.
  - user data will only be used for the functioning of the service and will in no way be stored in other databases in order to be correlated, aggregated or associated with other data sources for other purposes like building analysis models.
  - all information relating to the use of EMVE will be anonymized, eliminating any reference to the user who generated it.
  - the anonymized information can be used to perform any type of analysis deemed appropriate for the purpose of monitoring and improving the SocialTruth platform.

- **Data minimization:**
  - the digital companion will transmit to EMVE only the data strictly necessary for powering the engine itself, in order to provide the user with feedback on the information to be verified: no data, unless strictly necessary for the functioning of the platform, will be transmitted;
  - the most sensitive data processed is the url sent by the user. The url to be verified allows in theory to know what the subject's interests are. By associating the same user with a series of urls sent for verification, in fact, it would be possible to build a very detailed

profiling of the subject. Even more, considering that the professional end users of the platform (such as journalists and teachers) will be subjects, the danger of building analysis models related to their behavior is clearly evident. Think, for example, of the risk of being able to verify and monitor the urls visited by a particular journalist at a certain moment, who may be working on particularly delicate investigations; that said, the data processing structure will not allow associating n searches to the same user, even for a limited period of time. In fact, the URL data sent will be completely released from any information that can be traced back to the individual user, if not for the period strictly and technically necessary to provide the service to the user. Therefore, the platform will not store the user's IP address or any other data suitable for the identification

- **Accuracy:**
  - given the limited storage of data relating to users made by the SocialTruth platform, there are no particular critical issues regarding the principle of data accuracy. In compliance with the provisions of the GDPR, however, a contact point dedicated to users will be set up, dedicated to providing information and carrying out any data correction or update operations.

- **Storage limitation:**
  - the data necessary for the transactions between the user interface and the SocialTruth platform will be kept only for the period strictly necessary for the provision of the technical verification service. In no way will they be kept for a longer and unnecessary period; the deletion of the user data occurs when the Expert Meta Verification Engine sends the response to the digital companion.
  - The data deletion mechanism must be automated by the scripts dedicated to this function. It is necessary to set up a verification and auditing procedure that controls the effective deletion of data.

- **Integrity and confidentiality**:
  - in order to guarantee the integrity of the data, suitable measures must be taken to avoid accidental loss, destruction or damage of the data. A business continuity and disaster recovery plan must be prepared, and for this purpose suitable data backup processes must be adopted both on remote platforms and in physical places. Data backup must be performed automatically and with a frequency suitable to minimize the risks of continuity of the service (minimum daily cadence). The data recovery plan must be detailed, and the recovery time must be suitable to guarantee the restoration of the functionality of the service within a maximum time of 8 hours. The whole plan must be subjected to validation and verification tests carried out by an external expert;

    established that the storage of user data is limited to the minimum necessary; it is however essential to take into account the risks in relation to the confidentiality of the data: for this purpose, it is necessary to adopt all the appropriate measures that prevent

unauthorized access to information. For this purpose, it is necessary to configure the systems in such a way as to avoid unauthorized access. From an organizational point of view, it is necessary to appoint a system administrator who defines the user access policies to the platform and implements the related security measures and data access policies.

Therefore, in practical terms the following aspects should be taken into account:

- the data of the user is never stored anywhere (neither IP address, nor location etc.)
- the data from user(s) is never used to build any models/aggregated data (purpose limitation)
- no sensitive data is being used (GPS etc.)
- user can choose if and what will be stored in blockchain
- no one else can see the analysis/results of single user (unless desired).

In addition to the above, best practices will be adopted to ensure adequate level of security:

- all data communication channels will be encrypted
- if passwords are used, the password policy should be compliant with NIST updated guidelines
- before entering into production, it is recommended to perform SocialTruth platform penetration testing performed by an external provider.

4 Socio-technical and human aspects

This section is dedicated to the final refinement of the SocialTruth sociotechnical aspects. The main recommendations and requirements from the socio-technical perspective have been already extensively discussed in previous deliverables D2.1 and D2.2, and there is no need to recall them here as they were well received from all the involved work packages.

In this section we would like to address the following specific issues:

- the human aspects related to software architecture and software engineering (section 4.1);
- the role of cognitive biases in the spread of false information (section 4.2).

The other ethical issues concerning responsibility, gender management and data management will be addressed in deliverable 1.3.1 and 1.3.2 scheduled at M18 and M36.

4.1 Socio-technical considerations on software architecture design

Software Architecture may be defined as the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, the relations among them, and the properties of both the elements and relations[2]. The architecture of a software system is a metaphor, analogous to the architecture of a building[3]. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the design teams[4]. Software architecture is referred to as the notion of the most important aspects of the internal design of a software system[5].

Software architecture is a unique and highly complicated engineering discipline with fundamental cognitive, organizational, and resource constraints. These constraints are inherent due to the architecture's intangibility, intricate inner connections, the cognitive difficulty of software and their dependency on systems, diversity, and human. One may argue that the core elements of a software architecture are purely technical; nonetheless, some psychological aspects also exist that deserve a dedicated mention. The most disruptive psychological aspect is the one related to the human capability to understand complexity, coupled with the need to work together. It is paramount to understand and work with the human brains that realise the software, and the ones that eventually use it.

For this reason, the software architecture has to be relatively simple to understand. All assumptions must be challenged in order to find the path of the least resistance and reach an acceptable level of simplicity, so that all the people working to develop the software system can understand it. As described in section

[2] Clements, Paul; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford (2010). Documenting Software Architectures: Views and Beyond, Second Edition. Boston: Addison-Wesley. ISBN 978-0-321-55268-6

[3] Perry, D. E.; Wolf, A. L. (1992). "Foundations for the study of software architecture" ACM SIGSOFT Software Engineering Notes. 17 (4): 40.

[4] "Software Architecture". www.sei.cmu.edu

[5] Martin Fowler - Software Architecture Guide (2019)

2, the architecture design proposed in this document consists in well-defined system building blocks with clear connections and mutual relationships.

Then, the software architecture must contain fair rules telling people how to work together, considering the individual responsibility over the implementation of unit components and the encouragement of diversity of thought and action. There should be space and opportunity for an individual to perform acts that may benefit the whole, while keeping adherence to the rules for remixing and ownership[6]. In fact, the microservices contemplated in the design of the SocialTruth software system adapt the single responsibility paradigm, as described in section 2.1.

On the other hand, the architecture should also make silent experimentation easy, giving different teams the opportunity for success without punishing the failure encountered on the research road. This is particularly true for the different verification services to be developed, that fall within the area of research and, in fact, have a detailed lab-verification plan[7].

Finally, the software system architecture also has to give economic incentive to invest in making it happen: in this perspective, the architecture is also a market place. Indeed, the design described in this document and the lab-verification plan are both heavily driven by the end-user requirements gathered during previous phases of work package 2, that are continuously updated and refined, and by the overall exploitation plan[8].

With regard to the quality of the software architecture proposed here, it is important to recall that quality is a generic measure of the degree of excellence of a product or service against a given standard, and for software this measure is a multifaceted attribute characterizing the quantity of both utility and durability of the product and/or service.

Software quality can be perceived from a relative point of view as the conformity of a software system to its specifications (design models). Therefore, software quality is inversely proportional to the differences between the behaviours and performance of a software system and those required in the specifications. However, many quality attributes of software, such as design quality, usability, implementation efficiency, and reliability, cannot be quantified[9] and some qualitative or informal validation and evaluation techniques, such as review and prototyping, are adopted in software engineering[10]. These aspects are reflected into a certain degree of freedom that is left open in the software architecture, enabling agility and helping to embrace and implement changes in requirements or processes. Thus, some characteristics of a good architecture relate to good modularity reached through an appropriate decomposition strategy,

6 Pieter Hintjens "ZeroMQ Messaging for Many Applications" Chapter 6 Publisher: O'Reilly Media (March 2013) http://hintjens.com/blog:8

[7] SocialTruth deliverable D5.1 "Overall Evaluation Plan" (submitted on M14)

[8] SocialTruth deliverable D6.8 "Preliminary SocialTruth Business Plan" (submitted on M12) and upcoming updates

[9] Yingxu Wang, Shushma Patel "Exploring the Cognitive Foundations of Software Engineering" Int. J. of Software Science and Computational Intelligence, 1(2), 1-19, April-June 2009

[10] Jonathan Arnowitz, Michael Arent and Nevin Berger "Effective Prototyping for Software Makers" A volume in Interactive Technologies (2007) Elsevier Inc. https://doi.org/10.1016/B978-0-12-088568-8.X5000-0

irrespective of whether the solution under development has a monolithic or a microservices architecture[11].

4.2 Cognitive biases and the spread of false information online

In deliverable D2.1 we explained that "*falsehood diffuses significantly faster, deeper and more broadly than true news, especially regarding politics. It appears that false news is generally more novel, and that novel information is more likely to be shared, possibly because people feel more compellent about sharing novel news. Moreover, the emotional reactions of recipients of false news were found to be mainly surprise and disgust, whereas the truth inspired sadness, anticipation and trust. Also, the greater likelihood of people to retweet falsity more than the truth is what drives the spread of false news, despite network and individual factors that favour the truth. The recommendations about misinformation-containment policies include emphasizing behavioral interventions, like labelling and incentives to dissuade the spread of misinformation*."[12]

These are the most authoritative considerations that are currently available about the reasoning behind the spread of disinformation and falsehood online, but other aspects may contribute to this as well. Among them, we would like to investigate here the effect of **cognitive biases** affecting human reasoning.

A cognitive bias is a systematic pattern of deviation from objectivity and rationality in reasoning and judgment, that can be useful in everyday life, but can also sometimes lead to the distortion of reality, inaccurate judgment, illogical interpretation, or what is broadly called irrationality. As described by Tom Stafford, a senior lecturer in psychology and cognitive science at the University of Sheffield[13], *"Cognitive biases exist for very good evolutionary reasons. They are not rogue processes which contaminate what would be otherwise intelligent thought: they are the foundation of intelligent thought. Human beings must make decisions with limited time, information and intellectual energy, and useful short-cuts may be based on cognitive biases."*

As an example, let us consider the confirmation bias, which is the tendency to search for or interpret information in a way that confirms one's preconceptions. As Stafford explains, *"although there are risks to preferring to seek information that confirms whatever you already believe, the strategy does provide a way of dealing with complex information, and a starting point (i.e. what you already suspect) which is as good as any other starting point. It doesn't require that you speculate endless about what might be true, and in many situations the world (or other people) is more than likely to put contradictory evidence in front of you without you having to expend effort in seeking it out. Confirmation bias exists because it is an efficient information seeking strategy – certainly more efficient than constantly trying to disprove every aspect of what you believe*".

Many features of the human brain evolved in order to allow fast and energy-saving reactions to external stimulations. This is a very useful and important capability, which for example allows to promptly react to

[11] George Fairbanks "Just Enough Software Architecture: A Risk-Driven Approach" Marshall & Brainerd (2010) ISBN 10: 0984618104 ISBN 13: 9780984618101
[12] S. Vosoughi, D. Roy e S. Aral, "The spread of true and false news online" Science, vol. 359, pp. 1146-1151, 2018.
[13] Stafford, T. (2015) "Bias Mitigation"

dangerous situations. Neuroscientists explain the physiological mechanism for this fast reactivity in terms of differentiated processing of information within the brain (LeDoux model), based on parallel transmission of information from the thalamus to the amygdala (raw information, rapidly sent) and to the brain cortex (more complete information, sent more slowly).

This is a useful mechanism for making fast decisions, but cognitive biases can skew judgement and may have some particularly pernicious effects on the spread of false news, especially online, since **on the web all operations are quick and nimble**.

We propose here a very brief description of the most common cognitive biases affecting human reasoning[14]:

- Confirmation bias: the tendency to search for or interpret information in a way that confirms one's preconceptions
- Representativeness: the tendency to classify based on the partial similarities to something typical, characteristic, representative, already known; to the typical stereotype image
- Availability heuristic: the tendency to estimate what is more likely by what is more available in memory, which is biased toward vivid, unusual, or emotionally charged examples
- Anchoring: the tendency to rely too heavily, or "anchor" on a past reference or on one trait or piece of information when making decisions
- Hindsight bias: the tendency, after an event has occurred, to see the event as having been predictable, despite there has been little or no objective basis for predicting it, prior to its occurrence
- Framing effect: the tendency to decide on options based on whether the options are presented with positive or negative connotations
- Focusing effect: the tendency to place too much importance on one aspect of an event, causing error in accurately evaluating its importance
- Law of small numbers: the tendency to estimate the features of a sample population from a small number of observations or data points
- Probability neglect: the tendency to ignore a small risk or give it too much rank
- Frequency bias: the illusion in which a word, a name or other thing that has recently come to one's attention suddenly appears "everywhere" with improbable frequency
- Information overload: too much information causes a problem in effectively understanding an issue or making decisions
- Denial: facing a fact or information too uncomfortable to accept leads to rejecting it, despite what may be overwhelming evidence
- Post-storm neurosis: danger of overreacting to circumstances having just had a severe event

[14] PYTHIA project D2.4 "Recommendations on how to improve the accuracy of technology foresight" available at http://www.pythia-padr.eu/web/guest/public-deliverables

- Group-thinking: risk that group members stick to common assumptions and views that are never questioned and challenged
- False analogy: two objects (or events) are shown to be similar. Then it is argued that since one of the two objects has a certain property, so the other object must also have it

Very often false news produced on purpose leverages these cognitive biases and tricks the readers' brains, in the same way advertisements titillate consumers.

Another important aspect to be considered in this frame is the existence of ideological polarization and the so-called "filter bubbles" that make people belonging to closed online environments (such as Facebook groups) be subjected to selective exposure on social media[15]. But can we state that cognitive biases of filter bubbles represent a fundamental contribution to the spread of false news?

Actually, although **some authors claim that biases make people more vulnerable to misinformation** spread by social media[16], in particular with regard to the news about politics, recent literature works suggest that this may not really the case: Pennycook *et al.*[17] researched this issue and found that **susceptibility to fake news is driven more by lazy thinking than it is by partisan bias** per se. The author states that reasoning, when performed, allows people to effectively differentiate the fake from real regardless of political ideology.

Therefore, the key point is to find a way to make people stop for a little, think and consciously decide if they believe or not in the news they are reading and if sharing it would provide benefit. In 2010 Cheng and Wu[18] investigated possible moderators of the framing effect and found that a significant attenuation of this bias occurred when the participants of the experiments were subjected to **weak/strong warning messages**. The magnitude of this attenuation was found to depend on the level of involvement of the participants in the task. Less involved participants were more susceptible to the framing effect than the more involved subjects.

Taking these research works as reliable and recalling the recommendation proposed on Science to "*emphasize behavioural interventions, like labelling and incentives to dissuade the spread of misinformation"*, we believe that the approach proposed by the SocialTruth project, based on **awareness rising and labelling of the news** that are likely to be untrustworthy or not, may be successful and provide a genuine and relevant contribution to the global struggle against the spread of falsehood online.

[15] Dominic Spohr "Fake news and ideological polarization: Filter bubbles and selective exposure on social media" Business Information Review Volume: 34 issue: 3, page(s): 150-160 https://doi.org/10.1177/0266382117722446
[16] Giovanni Luca Ciampaglia, Filippo Menczer, The Conversation US (June 21, 2018)
[17] Gordon Pennycook, David G. Rand, "Lazy, not biased: Susceptibility to partisan fake news is better explained by lack of reasoning than by motivated reasoning" Cognition, Volume 188, 2019, Pages 39-50, ISSN 0010-0277, https://doi.org/10.1016/j.cognition.2018.06.011
[18] F.-F. Cheng e C.-S. Wu, "Debiasing the framing effect: The effect of warning and involvement" Decision Support Systems 49, p. 328–334, 2010.

4.3 Considerations on the democratic approach proposed by SocialTruth

In this section we would like to clarify some points with regard to the **democratic and pluralistic** approach proposed by SocialTruth, aiming at developing a **distributed content verification solution**.

The fundamental idea behind this approach is that a distributed system should allow to provide trustworthiness scores about the news and highlight credible information without affecting the essential freedom of journalism, which is a pillar of democratic society.

The system is open and democratic, so anyone, in principle, could plug in their own verification service. Who is responsible if fraudulent services are added, by mistake or on purpose? If we only accept "certified" services, then the system is not open anymore. If we accept all services with no evaluation of their quality, then the whole system becomes untrustworthy.

These are interesting points and hit a core aspect of democracy: the need for a certain degree of self-regulation to maintain quality standards and provide credibility, while not closing the system.

An example of how this aspect can be handled can be found in scientific disciplines and is represented by the peer review process, where anyone can contribute to general knowledge providing his or her own work products, and each work product is evaluated by all the other contributors (peers). If the work is considered credible, relevant and well done, it is somehow accredited as valuable. Following this mechanism, the performances of contributors can be tracked, and it is possible to build a reputation score.

However, given the aforementioned priorities related to system openness and democratic representation of diverse verification methods, SocialTRUTH is putting specific emphasis on delivering a quality controlled solution that will support the decision making of the addressed stakeholders: journalists, teachers and tutors, consumers and the general public. The verification services will be quality controlled and accompanied by validation methods and records before they will be opened to the public. Developers' details, tools, methods and certicicates will be documented and openly available to all end-users. Thus the systems will not be a hub for weakly justified and dubious verification efforts, but rather and open ecosystem where the most accurate and trustworthy services will be showcased and promoted. In any case, verification engines that would be community created and pose difficulties during QC and validation, will be accompanied with clear disclaimers of their status.

4.4 Considerations on the responsibility of the results and mistakes generated by SocialTruth platform

Further in the project, also the aspects of responsibility (ethical and legal considerations) of the possible mistakes should be analyzed.

As in all IT systems, some mistakes can be generated by using SocialTruth enabled platform and services, too.

Some examples are:

1. someone writes a false story about a known person (e.g. corruption or drugs) and chosen EMVE suggests it is truth - such a person can lose reputation, job, elections etc.
2. author/journalist writes a true story, and EMVE says it is fake...it might also cost reputation...

Who would be responsible and attributed for such mistakes?

As described in above sections, there might be several EMVE available to check the content.

Such an approach is very democratic and allows users to make many analysis and independent checks, and to choose the EMVE and services they trust.

Of course, different EMVE can be owned by private or public organizations (TV, portal, ministry, press agency, NGOs), and then those organizations take responsibility for the offered results (calculated according to their business model).

A key point that should be worth noting is that SocialTRUTH and its supporting end-user application, the Digital Companion, will aim at providing the end-user with a barometer or heatmap of the credibility of a selected source. The more verification services rank a selected source high enough, the more the reputation and trustworthiness of the source will be secured. The set of tools that will be developed by SocialTRUTH will not only strengthen the verification capacity of the stakeholders, but will also formulate a baseline of trusted sources that can be used to gauge and eliminate fake-news, misinformation and disinformation incident in the future.

4.5 Ethical and societal aspects within SocialTruth architectural design

SocialTruth architecture has been created by following Privacy by Design principle and within ethical and democratic considerations at start.

In particular, the following architectural choices implement ethical requirements as well as societal and democratic values:

1. Following open and democratic approach, SocialTruth architecture allows for having many EMVEs. It means, there is not one single-truth authority, but users can choose freely which EMVE to use, or they can compare results by using several EMVEs. In such ecosystem, each EMVE owner would and should be willing to offer correct and trustworthy results for citizens.
2. Following open and democratic approach, SocialTruth architecture allows for deploying many verification services. Each EMVE can use several services that could be offered by EMVE owners, but also by researchers, scientists, communities in a very open ecosystem. Moreover, users have the possibility to choose which verification services (offered by certain EMVE) they wish to use, and users are able to configure some settings of those services.

3. The results of the platform are verifiable and the results are possible to be re-calculated and checked by interested communities. All the results coming from the chosen verification services contain information about the settings and configuration parameters, so that the same input data (e.g. the link to the article/news to be checked) can be re-calculated by other actors (e.g. interested researchers or data scientists) with the same settings.
4. The results of verification will be stored in distributed blockchain.
5. SocialTruth platform is open and transparent in principle, allowing for creation of EMVE, data models, algorithms and verification services.

5 Conclusions

This document is the final release of the SocialTruth architecture design. It is the refined version of D2.2 document in which initial specification of the platform architecture has been prepared at the early stage of the project (until M6).

This report will help particular work packages' and technology/components' creators to follow the architectural principles and guidelines as well as common understanding of the SocialTruth ecosystem. As the input, we have used the requirements coming from D2.1, the best practices and knowledge of current technologies and architecture design principles, as well as findings, considerations and initial output of D2.2 deliverable.

This deliverable provides the logical and technical views on the SocialTruth platform, including blockchain-enabled distribution environment, information flow, aspects of microservices integration adopted to be used in the SocialTruth, aspects of data modelling, system monitoring and observability.

The report also displays the modules and components of the platform, lists the needed functionalities, discusses the interoperability aspects, as well as the security, privacy, social and human aspects.

At the time of submission of D2.3, the part of architecture (e.g. Kafka) is operationalized. We have developed the first prototype of the framework allowing for execution of two verification services (text based). Further development efforts are ongoing.

Some papers documenting our work on the architecture with initial results are now submitted (the results are not yet known), such as:

- Distributed Architecture for Fake News Detection for CISIS 2020 (authors from UTP)
- Fake News Detection from Data Streams for IJCNN 2020 (authors from UTP)

Detailed specification of platform modules, including technical aspects of their implementation will be provided in the later phase of the project in respective deliverables. Blockchain design and implementation, Lifelong learning expert system and Digital Companion module will be presented in details in D4.1-D4.3 documents, Deep Learning Multimedia Verification block in D3.4-D3.5, while integrated prototype of the platform in three releases of D5.2.x deliverables.

6 References

A., W. (2009). *The Colour Affects System of Colour Psychology.* AIC Quadrennial Congress.

Alicke, M. D., Vredenburg, D. S., Hiatt, M., & Govorun, O. (2001). The "better than myself effect". *Motivation and Emotion, 25*, 7-22.

Backholm, K., Ausserhofer, J., Frey, E., Grondhal Larsen, A., Hornmoen, H., Hogvag, J., & Reimerth, G. (2017). Crises, Rumours and Reposts: Journalists' Social Media COntent Gathering and Verification Practices in Breacking News Situations. *Media and COmmunication, 5*(2), 67-76.

Brandtzaeg, P. B., Lüders, M., Spangenberg, J., Rath-Wiggins, L., & Følstad, A. (2016). Emerging Journalistic Verification Practices Concerning Social Media. *Journalism Practice, 10*(3), 323-342.

Demangeot, C., & Broderick, A. J. (2007). Conceptualising consumer behaviour in online shopping environments. *International Journal of Retail & Distribution Management, 35*(11), 878-894.

Diakopoulos, N., De Choudhury, M., & Naaman, M. (2012). Finding and assessing soccial media information sources in the context of journalism. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* Austin, TX.

Djamasbi, S., Siegel, M., & Tullis, T. (2011). Visual Hierarchy and Viewing Behavior: An Eye Tracking Study. Human-Computer Interaction. *Design and Development Approaches in Computer Science*, 331-340.

Elder, R. S., & Krishna, A. (2012). The “Visual Depiction Effect” in Advertising: Facilitating Embodied Mental Simulation through Product Orientation. *Journal of Consumer Research, 38*(6), 988-1003.

Fadeyev, D. (n.d.). *The Usability Post - Thoughts on design and user experience*. Retrieved from http://usabilitypost.com

Gorini M., C. V. (2013). *EMERALD deliverable 'D2.3 - EMERALD System Functional Architecture'.*

Hallock, J. (2003). *Colour Assignment - Preferences and Associations.*

Imtiaz, S. (2016). *The Psychology Behind Web Design.* McMaster University.

Lindgaard, G., Fernandes, G., Dudek, C., & Brown, J. (2006). Attention web designers: You have 50 milliseconds to make a good first impression! *Behaviour & Information Technology, 25*(2), 115-126.

Schifferes, S., Newman, N., Thurman, N., Corney, D., Goker, A., & Martin, C. (2014). Identifying and verifying news through social media: developing a user-centered tool for professional journalists. *Digital Journalism, 2*(3), 406-418.

Schwartz, R., Naaman, M., & Teodoro, R. (2015). Editorial algorithms: using social media to discover and report local news. *Ninth International AAAI Conference on Web and Social Media.* Oxford, UK.

Sillence, E., Briggs, P., Fishwick, L., & Harris, P. (2004). Trust and mistrust of online health sites. *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, 663-670.

StatCounter. (n.d.). Retrieved from http://gs.statcounter.com/

The Open Group. (n.d.). *ArchiMate®, 2.1 specification*. Retrieved December 2013, from The Open Group: http://pubs.opengroup.org/architecture/archimate2-doc/

Tuch, A. N., Presslaber, E. E., Stöcklin, M., Opwis, K., & Bargas-Avila, J. A. (2012). The role of visual complexity and prototypicality regarding first impression of websites: Working towards understanding aesthetic judgments. *International Journal of Human-Computer Studies, 70*(11), 794-811.

Verhagen, T., Boter, J., & Adelaar, T. (2010). The Effect of Product Type on Consumer Preferences for Website Content Elements: An Empirical Study. *Journal of Computer-Mediated Communication, 16*(1), 139-170.

Veromann, V.-J. (n.d.). *RAG+B traffic light rating system – expanding established design patterns.* https://weekdone.com.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering* (pp. 195-216). Springer, Cham.

Dragoni, N., Lanese, I., Larsen, S. T., Mazzara, M., Mustafin, R., & Safina, L. (2017, June). Microservices: How to make your application scale. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics* (pp. 95-104). Springer, Cham.

Microservices Authentication and Authorization Solutions, available online: https://medium.com/tech-tajawal/microservice-authentication-and-authorization-solutions-e0e5e74b248a (accessed May 23, 2018).