Grant Agreement No.: 761488

D3.2: Technology Bricks V1

This deliverable provides the first version of the technology bricks which have been planned for the first prototype of the CPN platform. The prioritization of the features to implement for the first prototype have been carried out in such a way as to respect the expected timing, but at the same time release meaningful functionalities, that will be improved and extended in the next prototypes.

Work package	WP 3
Task	T3.1-T3.3
Due date	30/6/2018
Submission date	30/6/2018
Deliverable lead	ATC
Version	1.0
Authors	Nikos Sarris, Marina Klitsi, Efstratios Tzoannos, Stamatis Rapanakis (ATC)
Reviewers	Fulvio D'Antonio (LiveTech)
Keywords	Technology Bricks, APIs, prototype 1

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.1	11/5/2018	Table of Contents	Nikos Sarris, Marina Klitsi (ATC)
V0.2	25/6/2018	1st completed version including partners' contribution	Nikos Sarris, Stamatis Rapanakis, Marina Klitsi (ATC), Michele Nati (DIGICAT), Bosco Ferdinando (ENG), Fulvio D'Antonio (LiveTech)
V0.3	27/6/2018	Review of the report	Fulvio D'Antonio (LiveTech)
V1.0	28/6/2018	Final version of the reportt	Nikos Sarris, Stamatis Rapanakis, Marina Klitsi (ATC)

Co-funded by the Horizon 2020 Framework Programme of the European Union

DISCLAIMER

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 761488.

This document reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		**R**
Dissemination Level		
PU	Public, fully open, e.g. web	**X**
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to CPN project and Commission Services	

Co-funded by the Horizon 2020 Framework Programme of the European Union

EXECUTIVE SUMMARY

This deliverable reports on the work performed in WP3 which addresses the development of the required technology bricks for the CPN Platform. Taking into account the user requirements, as described in deliverable D1.1 "User Requirements Model", this report presents the components and services implemented for the first prototype of the platform.

The components & services which are being presented in this deliverable are classified in three main categories (Content, Users, Mapping), based on their functionality as defined by the project's requirements. For each technology brick, a brief description of its functionality is provided, along with API, test scenarios and installation guidelines.

The first prototype of the CPN platform includes the following 6 technology bricks, as described in this deliverable:

Layers	Name of 'technology brick'
Content Technology Bricks	Relation Extraction
Users Technology Bricks	User Modelling
	Reader's App
	Personal Data Receipts
Mapping Technology Bricks	Producer's App
	Recommender

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020 Framework Programme of the European Union

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

API	Application Programming Interface
PDR	Personal Data Receipts
App	Application

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

This Deliverable contains the basic description of the technological infrastructure of the 1st prototype of the CPN platform which is composed by what we call 'technology bricks'. The components, APIs, and services included in the first version of the platform customization infrastructure and components, and described in this deliverable have been designed and developed according to the user requirements, as described in deliverable D1.1 “User Requirements Model”.

The CPN project foresees three releases of the 'technology bricks' in order to be available for the related pilots. Each release includes specific functionalities, chosen after a process of evaluation and prioritization of the user requirements. Starting from the reference architecture document, the first versions of the technology bricks have been implemented and reported in this document. This version of the bricks is the first one of a cycle of three iterations and will offer a series of features in order to test the related bricks in a pilot environment.

The main goal of this document is to present the technology bricks that are foreseen at this point of the project necessary to satisfy the user requirements expected for the first pilot iteration. For each technology brick, a brief description of its functionality is provided, along with API, test scenarios and installation guidelines.

The structure of the deliverable is organized as follows: Section 2 provides an overview of the requirements for the 1st prototype. Section 3 describes the first versions of the CPN technology bricks infrastructure. Finally, section 4 concludes this document.

Co-funded by the Horizon 2020
Framework Programme of the European Union

2 SUMMARY OF REQUIREMENTS FOR PROTOTYPE 1

We enlist here the requirements that we have satisfied in the first prototype along with the list of the modules that have been necessary for satisfying these requirements.

PROTOTYPE 1

Requirement category	Requirement ID	Relevant Task	Requirement description
UR-UP1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.2	API Gateway must expose API to collect user actions User Modelling must create/refine user interests Reader's App must track user actions	The system should create/refine interests based on the user's consumption habits
	UR-UP 1.4	Recommender - Refine user's interests through user interaction Reader's App - Support interaction to refine user's interests (talkback). Reader's app must refine user's interest through user interaction.	The system should refine the user's interests through frequent interaction with the user (talkback)
	UR-UP 1.6	User Modelling must assign preferences to categories after collecting user actions	The system should assign preferences (1-5) to categories based on the users behaviour
	UR-UP 1.7	Recommender must allow users to assign and change preferences (1-5) to categories themselves. Reader's app must allow users to assign and change preferences (1-5) to categories themselves.	The system should allow users to assign and change preferences (1-5) to categories themselves
	UR-UP 1.8	Recommender should expose API to enable/ disable personalisation algorithm. Reader's App should contain UI elements to enable/ disable personalisation algorithm	The system must allow users to completely turn off the personalisation algorithm and receive content as is and vice versa
UR-UP2: Network: Making use of connections the user already has through social media.	UR-UP 2.7	Reader's app should allow users to share content to social networks.	The system should allow users to share content from the CPN system to social networks
UR-UP3: Time & Length:	UR-UP 3.1	Reader's app must allow the user to choose a preferred time frame	The system must allow the user to choose a

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Relevant Task	Requirement description
When does the user prefer to consume content and for how long?		or frames to consume content, to postpone it and to ignore it. Recommender must allow the user to choose a preferred time frame or frames to consume content, to postpone it and to ignore it.	preferred time frame or frames to consume content
	UR-UP 3.2	User Modelling must track user consumption habits.	The system should create/refine time frames based on the user's consumption habits
	UR-UP 3.3	Recommender must refine the user's time frames through frequent interaction with the user (talkback) Reader's App - Support interaction to refine the user's time frames (talkback). Reader's app should support refining the user's time frames through frequent interaction with the user (talkback)	The system should refine the user's time frames through frequent interaction with the user (talkback)
	UR-UP 3.5	Reader's app must allow the user to choose a preferred time frame or frames to consume content, to postpone it and to ignore it. Recommender must allow the user to choose a preferred time frame or frames to consume content, to postpone it and to ignore it.	The system must allow the user to postpone a time frame for a chosen amount of time.
	UR-UP 3.6	Reader's app must allow the user to choose a preferred time frame or frames to consume content, to postpone it and to ignore it. Recommender must allow the user to choose a preferred time frame or frames to consume content, to postpone it and to ignore it.	The system must allow the user to ignore a time frame completely
UR-UP5: Location & Surroundings: Where is the user and what's	UR-UP 5.2	Recommender must be able to set a home/main interest location. Reader's App support setting a home/main interest location.	The system should allow the user to set a home/main interest location

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Relevant Task	Requirement description
going on around him/her?			
UR-UP9: User Profile Management: Giving the user transparency and control over their data	UR-UP 9.1	API Gateway must expose API to provide user data to CPN modules Reader's App should provide user data information and explanation Personal Data Receipts should provide an API to create/ edit/ retrieve a user's record	The system must provide transparent, simple and easy-to-understand information on what user data are collected, for what purpose and how they are stored
	UR-UP 9.2	API Gateway must expose API to provide user data to CPN modules Reader's App must offer management of personal user's data. User Modelling should provide user fields to Personal Data Receipts Personal Data Receipts must define the records fields	The system should require informed and explicit consent for processing of personal user data, beyond those required for the provisioning of the agreed service
UR-AF1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.5	Reader's app should allow users to choose favourite sources. Recommender app should allow users to choose favourite sources.	The system should allow users to choose favourite sources
UR-AF2: Avoiding FOMO: How to ensure people think they know everything there is to know	UR-AF 2.4	Recommender must show users only a limited number of items at once	The system should show users only a limited number of items at once
	UR-AF 2.5	Recommender must offer more content once all proposed articles have been consumed.	Once all articles proposed have been consumed, the system should only offer more content upon request by the users
UR-AF3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.4	Recommender should offer both news content and entertainment, locally and globally relevant content.	The system should be able to offer both news content and entertainment
	UR-AF 3.5	Recommender should offer both news content and entertainment, locally and globally relevant content.	The system should be able to offer both locally and globally relevant content
	UR-AF 3.8	Reader's app should allow users to filter content by language.	The system should allow users to filter content by language

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Relevant Task	Requirement description
		Recommender must allow users to filter content by language.	
UR-AF4: Sources: Where does the necessary content come from?	UR-AF 4.1	Recommender must elaborate contents provided by Producer's App	The system should be able to personalise news from/for the CPN media partners (VRT, DIAS, DW)
	UR-AF 4.2	Reader's app must allow for additional content sources, outside the consortium. Producer's App must allow for additional content sources, outside the consortium	The system should allow for additional content sources, outside the consortium
UR-AF7: User Feedback: Asking users to help improve the system	UR-AF 7.2	User Modelling should take into consideration user feedback. Reader's App support sending feedback, add labels next to UI elements	The system should include guided feedback for specific elements of the system, allowing users to (help) improve it

Table 1: First prototype requirements

Foreseen necessary technology bricks

The technology bricks necessary for the 1st prototype are illustrated in the shaded cells of the table below.

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Sentiment
Users Technology Bricks	User Modelling
	Reader's App - TRULY MEDIA
	Personal Data Receipts
Mapping Technology Bricks	Producer's App - CUTE4LE
	Reward Framework
	Twitter Analytics - TRUTHNEST
	Recommender

Table 2: First prototype technology bricks

Co-funded by the Horizon 2020 Framework Programme of the European Union

3 MODULES DESCRIPTION

This section describes the modules needed to implement the 1st prototype of the CPN platform.

3.1 RELATION EXTRACTION

3.1.1 Overview

The relation extraction module extracts entities from raw text and detects whether a predefined set of relations occurs between them. Our current system is designed for relations defined in the TAC KBP Shared task schema http://surdeanu.info/kbp2014/TAC_KBP_2014_Slot_Descriptions.pdf. We intend to allow for extension of this set of relations, depending on requirements of the recommendation system.

The module will process English text provided by the CUTE module. Given that named entities are essential for the slot filling component, next to relations, output from the named entity extractor will be included as additional metadata.

3.1.2 Role

Output from the relation extraction module will provide articles with additional metadata to be used by the recommender system to improve recommendations.

Articles are retrieved from CUTE module.

Articles with additional metadata on relations and entities are sent forward to the recommender module

3.1.3 Internal architecture

The module is a Python wrapper around the Stanford CoreNLP toolkit for java which provides a broad set of language technology tools.

The Python scripts post-process output from a CoreNLP server and will allow for detection of additional relations (not included in the the TAC KBP schema). This functionality relies on python packages such as Scikit-Learn, Pytorch,

3.1.4 API

The API of this module will provide a single operation, which is relation extraction and named entity recognition. Currently the module expects a single article for each call to the module and returns relations and entities detected at the sentence level.

Example of expected Input:

```
{
        _id : String, // Internal Id
        originId: String, // Original Id from source
        origin: String, // Source
        url: String //original item url
        category: String, // Category of article
```


Co-funded by the Horizon 2020
Framework Programme of the European Union

```
        title : String,
        text : String,
        language : String,
        author : String,
        date : Date, // date in Date format
        dateStr : String,    // date in String format
        timestamp : Number, // date in timestamp format
        location : { //location in geoJson format
                type: Object,
                index: '2dsphere',
                sparse: true },
        tags : [String] //list of tags
}
```

Example of output:

```
{
        originId : String, // Original Id from source
        origin : String, // Source
        url : String //original item url
        title : String,
        language : String,
        author : String,
        date : Date, // date in Date format
        dateStr : String,    // date in String format
        timestamp : Number, // date in timestamp format
        sentences: { [ {
                        index: int,
                        entitymentions: [ {
                                        CharacterOffsetBegin: int,
                                        CharacterOffsetEnd: int,
                                        DocTokenBegin: int,
                                        docTokenEnd: int,
                                        text: text // Surfaceform of Entity, eg. Barrack Obama
                                        ner: text //Nertype, eg. PERSON
                                        tokenBegin: int,
                                        tokenEnd: int
                                        }
                                        ],
                        kbp: [ {
                                        Object: string, // Object Entity, eg. Barrack Obama
                                        objectSpan: [int,int],
```


Co-funded by the Horizon 2020 Framework Programme of the European Union

```
                                relation: string, // Object Entity, eg. per:title
                                relationSpan: int,
                                subject: text // Object Entity, eg. President
                                subjectSpan: [int,int]
                        } ]
                } ] }
}
```

3.1.5 Test scenarios

During development articles were sampled from the API provided by Deutsche Welle and processed. Depending on the functionality (the addition of extra relations vs. the standard KBP schema mentioned earlier) the module has a latency in the order of 0.1 second per article. In subsequent tests we have processed a batch of articles from the CUTE module.

3.1.6 Installation and administration guidelines

The module can be easily deployed as a docker container using standard container orchestration technologies.

3.2 USER MODELLING

3.2.1 Overview

The user modelling module is responsible for creating, maintaining and building a profile of the users of the CPN Reader's App. In addition it is also responsible for analysing the articles collected by the Cute4LE module by performing a "semantic enrichment" of the text collected; the multilingual text is extracted in order to find relevant entities (such as person names, organizations names, locations etc.) and extracting keywords.

3.2.2 Role

This module constitutes one of the most important elaboration points in the CPN processing pipeline as it lays down the basis for the recommendation module to work. The recommendation process can be seen as a matching process between a list of users and a catalogue of items. Users and items can be described ad vectors of features. The module is responsible for creating, maintaining and continuously updating such vectors of features associated to users and news items.

In this first release the features used to represent Users are the following:

- **Informational features**: name, email contact
- **Socio-demographic features**: age range
- **Behavioural features**: news reading history, like/dislike history
- **Topics of Interests**: the ranked of list of topics for which the user has implicitly or explicitly expressed interest for over the time

Co-funded by the Horizon 2020 Framework Programme of the European Union

These features constitute the "User profile". The user profile is built partly on the information provided explicitly by the user himself and partly on the information provided by the user in an implicit way (e.g. long time spent reading an article is considered an implicit manifestation of interest for that article and the related topics)

The features used to represent "News Items" are the following:

- **Basic article metadata**: URL, date, tags (if provided by the original source)…
- **Automatically extracted metadata**: named entities (persons, locations, organizations cited in the text), unsupervised list of keywords, relations between entities in the text.

The module is able to extract metadata from three different source languages, namely english, dutch and greek.

3.2.3 Internal architecture

The module relies on the following components:

- **Components for retrieving messages from a broker**

They are in charge of connecting to the kafka broker end retrieve different kind of messages: news elaboration messages (new articles to elaborate) and user events.

- **NLP processing components**

Elaboration pipelines able to extract meaningful information from multilingual text. The core technologies actually used are: NLTK, Spacy, Polyglot (python libraries) and Stanford CoreNLP (java library/server).

- **Storage components**

Storing the result of the elaborations and users personal data: the backends used in the storage module are MongoDB and Apache Solr search server.

- **REST API Layer**

A python Flask layer of services exposing the CRUD operations for user profile management.

3.2.4 API

This module is responsible for maintaining users' data keeping track of his/her interests, history of click and news consumption and demographic data. This data will be mainly used by the recommender engine in order to select the most suitable news in according to a given user profile. The data consumed by this module will be the list of events generated by the users (posted through the broker). The kind of events that the module will be processing are the following:

- **"News":** when a news item is added to the broker it is retrieved by the module in order to tag, analyze it, extract topics, etc.
- **"Users_feedback"**: every action initiated by a customer that is collected by the platform it is collected, processed and stored in the user profile. Typical events are:

Co-funded by the Horizon 2020 Framework Programme of the European Union

- **User clicks**
- **Users ratings** (explicit rates, thumbs up/down, etc.)
- **User profile update** (e.g. change of name, location, age, etc.)

Data produced

Schema Version 0.1:

userModellingSchema:

```
{
 "user_id": "String",
 "demo": {
  "gender": "String",
  "age": "Number",
  "name": "String",
  "email": "String"
 },
 "interests": [
  {
   "id": "String",
   "label": "String",
   "score": "Number",
  }
 ],
 "activities": [
  {
   "item_id": "String",
   "event": "String"
  }
 ]
}
```


Co-funded by the Horizon 2020 Framework Programme of the European Union

3.2.5 Test scenarios

The module has been tested in isolation and in a full integrated way receiving text from Cute4LE module via the Kafka broker, elaborating and storing into Mongo and SOLR search server. The pipeline is currently active and elaborating all the test documents that are submitted on the broker.

3.2.6 Installation and administration guidelines

The User Modelling module is composed of several submodules all deployable by using docker orchestration technologies (e.g. Cattle, Kubernets, etc.). It is easily deployable using CPN platform's orchestration dashboard (based on Rancher) and a docker-compose file with all the relevant settings has been provided.

3.3 READER'S APP

3.3.1 Overview

The front end of the CPN application is provided by the Reader's App. It is a web application that mainly displays a news stream and offers to the user several controls to interact with the platform. Apart from displaying the Recommender module output (as a news stream), it is used to customize the user personalization settings, to track the user's actions and to get feedback from the user. The Reader's App is the starting point of the CPN application and provides a subscription and login user interface. It exchanges information with the other modules through an API Gateway and displays their output in a responsive, mobile friendly user interface. Finally, it offers a feedback form to send user feedback directly to the administrators.

3.3.2 Role

The role of the Reader's App is to provide a web interface to the user so that he can interact with the CPN application. The user subscribes and logins to the application by calling the respective API Gateway services. The User Modelling module is called to create a user profile. The Recommender module is called when the user logs in and its output (list of articles) are displayed as a "News stream". The Reader's app also shows information propagated by the Personal Data Receipts module, related to the user's data and usage policy. The user can manage his personal data access and usage settings through the application.

To support content personalization functionalities, Reader's App web application contains settings pages for the Frame Based Slot – filling system module. The user can enter his preferences and the respective module settings are updated. Reader's App tracks several user actions and sends them to the user modelling module. Examples of tracked user actions include the articles read, the rating of an article, the removal of an article from the news stream, the time spent on certain pages and others. In this way, the Recommender module can update its output and provide personalized content. Three streams (Personalized/ Popular/ All) are available in the Reader's App main page. The user can also view a collection of read/ favourite articles, as well as a collection of articles they disliked/ are not interested in. By using the provided functionalities he will better understand the way the Recommender is set up.

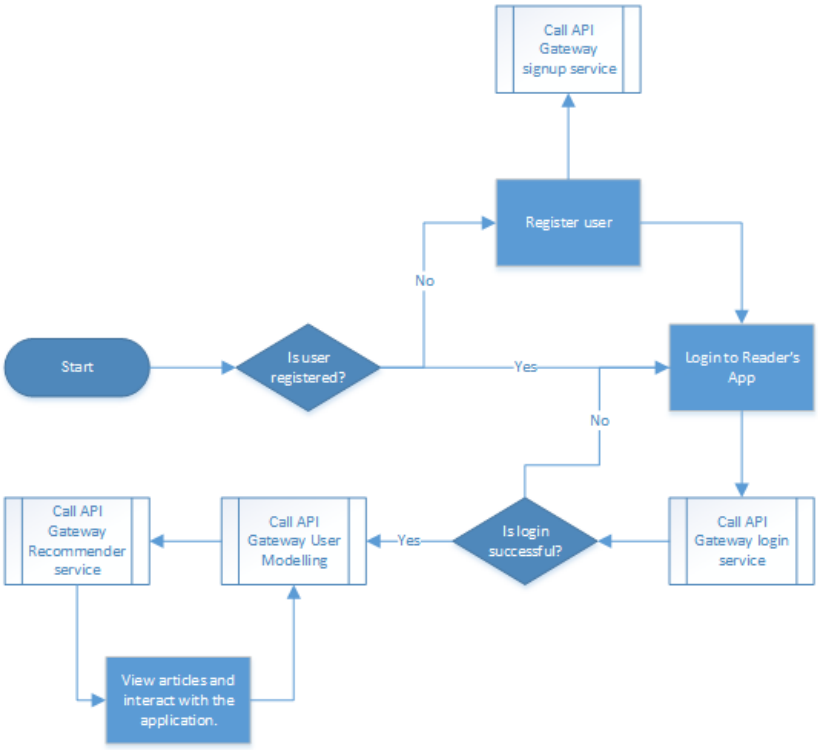
The following requirements that are addressed by this module in the 1st prototype are being considered complete: UP1.2, UP1.4, UP1.8, UP9.1, UP9.2 and AF4.1.

Co-funded by the Horizon 2020 Framework Programme of the European Union

3.3.3 Internal architecture

The Reader's App module consists of a front end web application and a backend Tomcat7 application. The backend application mainly manages the API Gateway communication (validations, error handling etc.) and the web application usage related data. It is developed using Spring Bootstrap framework (Java) and the data are stored in a Mongo database. The front end is based on Angular Fuse template (Angular 1.5.x) and communicates with the backend through RESTful web services.

The logical view of the modules and its main interactions with the other modules are shown in the following diagram:



Reader's App flow and interactions

The Reader's App backend has been developed to handle errors in a consistent and user friendly way. For example the modules error messages are being wrapped and more comprehensive messages are being shown to the user. A number of checks of the provided user input is being also performed (at the backend), to ensure the modules receive valid data.

3.3.4 API

Reader's App module does not offer an API since it is not called by another module. The interactions with the other modules are being performed through the API Gateway. Since it consists of a front end web application and a backend service, it implements an internal API for communication between them.

3.3.5 Test scenarios

The main test scenarios performed include the following functionalities:

- Subscribing to the platform.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Signing in to the platform.
- Rendering recommender module output (news streams).

Each of these actions contains the testing of success and error cases. The user input validation has been tested and the error messages of the modules are being wrapped to more comprehensive messages. The rendering of the recommender output requires special handling, as it can contain html code and its size should comply with the stream view limitations.

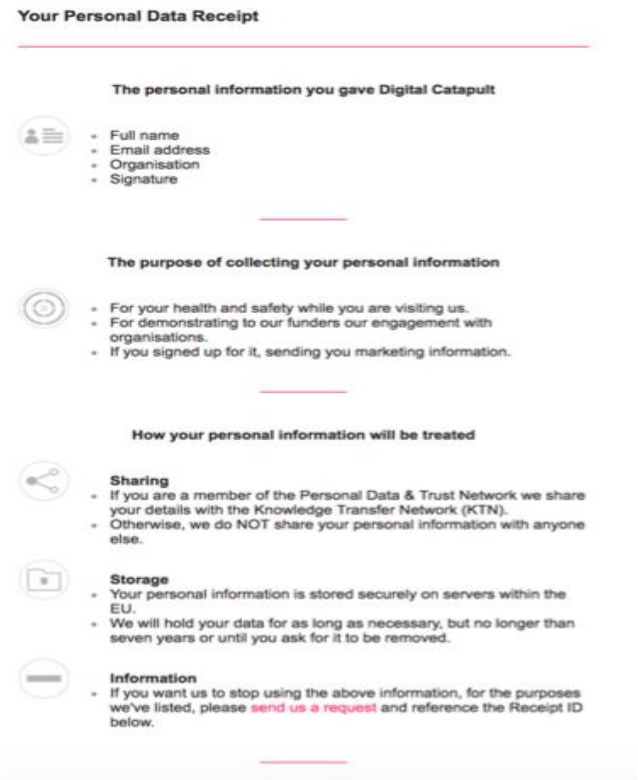
3.3.6 Installation and administration guidelines

The front end web application is deployed on Amazon CloudFront cloud service. The backend is deployed on an embedded Tomcat 7 on ATC premises. The user interface is self-explanatory. The main user interface controls offer extra information in the form of tooltips and illustrations.

3.4 PERSONAL DATA RECEIPTS

3.4.1 Overview

Personal Data Receipts (PDRs) are a way to simplify privacy policies by making them easier to understand for end-users and customers. Personal Data Receipts are implemented in the form of an email sent to a new user after registration is completed or to old ones when an update of previous privacy policies is performed. PDR emails implement a layered privacy policy approach, by providing a concise summary of what data are collected, how they are used, for what purpose, if third party sharing is provided and for how long such data are stored. They also provide a quick access to user rights over her data (including data removal). A mock-up of the PDR is provided below.



Your Personal Data Receipt

The personal information you gave Digital Catapult

- Full name
- Email address
- Organisation
- Signature

The purpose of collecting your personal information

- For your health and safety while you are visiting us.
- For demonstrating to our funders our engagement with organisations.
- If you signed up for it, sending you marketing information.

How your personal information will be treated

Sharing
- If you are a member of the Personal Data & Trust Network we share your details with the Knowledge Transfer Network (KTN).
- Otherwise, we do NOT share your personal information with anyone else.

Storage
- Your personal information is stored securely on servers within the EU.
- We will hold your data for as long as necessary, but no longer than seven years or until you ask for it to be removed.

Information
- If you want us to stop using the above information, for the purposes we've listed, please send us a request and reference the Receipt ID below.

Co-funded by the Horizon 2020 Framework Programme of the European Union

The Personal Data Receipts Creator is the module in charge of creating and sending PDRs in the scenarios identified above.

3.4.2 Role

Role of the Personal Data Receipts Creator is:

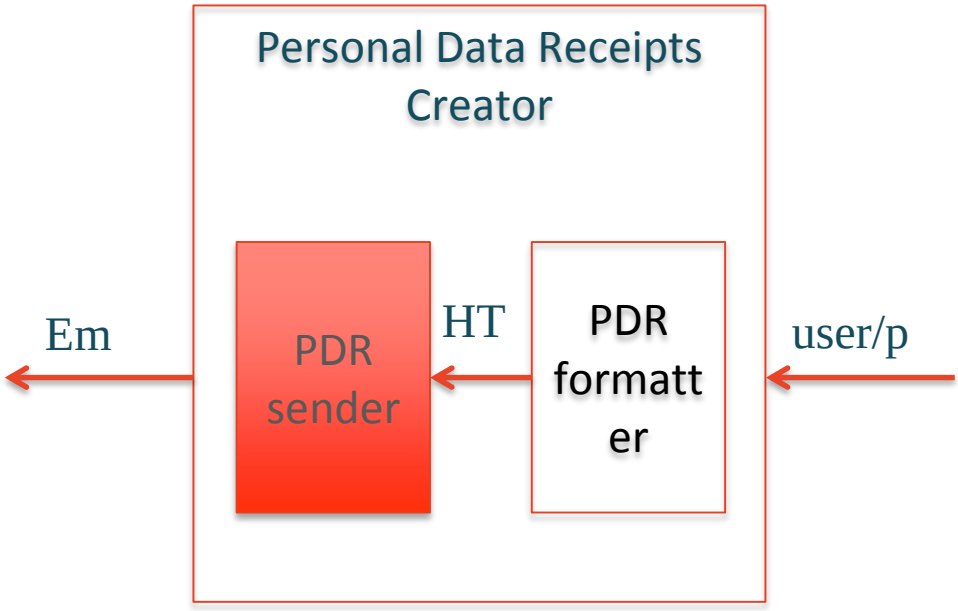
- Receive meta-information from the User Content Profiler in order to create a customised Personal Data Receipt for a given registered user. Collected meta-information will contain email address of the user, type of collected personal data, reason for collection and if third party sharing is involved, legal basis for data processing as well as user rights over her data and Data Controller contact details. A JSON representation of the exchanged meta-data is provided below.

```
{

    "name": "text",

    "email": "text",

    "profile_id": "string",

    "language": "text",

    "data_controller": "v_card",


    "user_data": [{

        "category": "text",

        "purpose": "text",

        "legal_basis": "text",

        "my_rights": "text",

        "start_date": "timestamp",

        "end_date": "timestamp",

        "retention_period": "number of days",

        "where": "text",

        "sharing": "boolean",

        "why": "text"

    }]


}
```


Co-funded by the Horizon 2020
Framework Programme of the European Union

- Create a customized Personal Data Receipt using the obtained meta-information to fill a pre-defined template
- Pass the created Personal Data Receipt in the form of an HTML file to a CPN email service or by pushing it back as notification through the ATC user app (not implemented in this phase).
- The provided implementation of the PDRs Creator will address the following requirements as detailed in D3.1: UR-UP9.1.

3.4.3 Internal architecture



3.4.4 API

- The module only provides an API call to generate the creation of a PDR. The following API /user/pdr POST is provided along with the following data model

```
{

    "name": "text",

    "email": "text",

    "profile_id": "string",

    "language": "text",

    "data_controller": "v_card",

    "user_data": [{

        "category": "text",

        "purpose": "text",

        "legal_basis": "text",
```


Co-funded by the Horizon 2020 Framework Programme of the European Union

```
            "my_rights": "text",
            "start_date": "timestamp",
            "end_date": "timestamp",
            "retention_period": "number of days",
            "where": "text",
            "sharing": "boolean",
            "why": "text"
        }]
}
```


3.4.5 Test scenarios

- User registration through the ATC user app
- Information on the created profile is extracted from the message bus and API call performed

/user/pdr POST


```
{
        "name": "Michele",
        "email": "Michele.nati@email.com",
        "profile_id": "QWERTY12345",
        "language": "English",
        "data_controller": "CPN, 101 Euston Road, NW1 2RA, ",

        "user_data": [{
            "category": "text",
            "purpose": "text",
            "legal_basis": "text",
            "my_rights": "text",
            "start_date": "timestamp",
            "end_date": "timestamp",
            "retention_period": "number of days",
```


Co-funded by the Horizon 2020 Framework Programme of the European Union

"where": "text",

"sharing": "boolean",

"why": "text"

}]

}

- HTTP/1.1 200 OK received on success
- Email received

3.4.6 Installation and administration guidelines

A Docker container.

3.5 PRODUCER'S APP

3.5.1 Overview

The producer's app is an evolution of Cute4LE which is a content curation platform for Marketing support using a storytelling approach. It is specialized for Large Events management. The platform allows to create stories reusing and embedding content, including user generated content harvested from web and social networks. User engagement mechanisms, live stream & Territory monitoring, influencers & trending topics management and analytics processes are blended together with the aim to exploit social network dynamics and monitor the activities related to specific events. The mechanisms cover different phases from pre-event to post-event.

3.5.2 Role

This module, in the 1st prototype, aims to provide the CPN platform with the articles extracted from media partners' repositories.

The requirements addressed by this module are:

- AF 4.1 : The system should be able to personalise news from/for the CPN media partners (VRT, DIAS, DW)
- AF 4.2 : The system should allow for additional content sources, outside the consortium

3.5.3 Internal architecture

- The producer's app consists of two docker containers: a customized image with the business logic and a MongoDB image for storage.
- This module retrieves in scheduled way the contents from DW, VRT and DIAS APIs, it stores these contents in a MongoDB collection, it publishes them to the message broker and it makes them available to the platform.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- The module has no external dependencies (without considering the APIs exposed by the media partners)

3.5.4 API

The producer's app exposes only an API to the CPN platform in this version of prototype. The API allows to retrieve a list of articles from and takes as input a list of articles ids. The goal of this API is to provide the contents to the reader's app, in base of the action of the recommender module. In fact, the recommender module, as output of its works, provides a list of articles ids that are used to retrieve the contents.

The API is exposed for internal use and is documented with OpenAPI specification (Swagger v2.0). The API is testable via API gateway interface and integration test that involve the orchestrator and recommender module is expected.

3.5.5 Test scenarios

A readers, through the reader's app, requires a personalized list of contents. The reader's app call the CPN API gateway and a recommendation process start. The orchestrator ask the recommender module a list of contents for the readers (user ID as input). The output of recommendation work is a list of articles ids that is passed as input to producer's app. The producer's app retrieve the list of contents from the storage and return them to the reader's app (via orchestrator).

3.5.6 Installation and administration guidelines

The module consists in two docker container. A docker-compose.yml is provided to build and push the service on CPN private registry.

The module exposes only an internal API and not requires administration operations.

The internal port is 8080.

3.6 RECOMMENDER

3.6.1 Overview

The module is in charge of computing the most suitable news recommendations for CPN users. It has to analyze the users' profiles and collected news to find the most "interesting" news items to be proposed by the app.

3.6.2 Role

The current state of the art techniques for recommending items are based on two main areas: content based (that relies on good semantic modelling/feature extraction and selection on the items to be recommended) and collaborative filtering techniques (that are essentially domain-independent and take into account network metrics based on emerging similarity graphs of users and items). Our system uses an hybrid approach that uses variable proportions of the mentioned techniques for each user learning (using Machine Learning techniques) from explicit and implicit feedback given by the users themselves: clicks, ratings, sharings, etc. The system is customizable for including content-delivery strategies' optimization: multichannel and date/time optimization (predicting the probability of interests at a given time on a given channel) and includes mechanisms for fostering "serendipitous" discoveries.

Co-funded by the Horizon 2020 Framework Programme of the European Union

The recommender exploits the features extracted from the document enriching modules: relation extraction, user modelling, user feedbacks, topic annotation.

3.6.3 Internal architecture

The module relies on the following components:

- **Components for retrieving messages from a broker**

They are in charge of connecting to the kafka broker end retrieve messages that will trigger a new recommendation for a specific user

- **Recommendation components**

Elaboration pipelines that are used to actually compute the recommendations for all the users: the technique used are content-based recommendations implemented relying on Apache Solr search server and NLP pipelines and "Collaborative filtering techniques" using custom and state of the art libraries such as python Surprise and Implicit.

- **Storage components**

Storing the result of the recommendation process: the backends used in the storage module are MongoDB and Apache Solr search server.

- **REST API Layer**

A python Flask layer of services exposing the CRUD operations for recommendation retrieval.

3.6.4 API

Data consumed

The data consumed by this module is the output of DS4Biz-UserModelling module and the output of the news and social media collector modules

Data produced

We will start with a very simple schema for the recommendations to be updated in the next releases.

Model schema:

[{

id : String, // Internal Id of the recommendation

user_id : String, // Internal Id of the user

score : Number, // The relevance score of this recommendation as computed by the recommender engine

date : Date // When this recommendation was computed

}]

Examples

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
[
  {
    "id":"xxxxxx",
    "user_id":"yyyyyyy",
    "item_id":"zzzzzzz",
    "date":"Fri, 11 May 2018 08:40:10 +0000",
    "score":0.8
  },
  {
    "id":"xxxxxx",
    "user_id":"yyyyyyy",
    "item_id":"zzzzzzz",
    "date":"Fri, 11 May 2018 08:40:10 +0000",
    "score":0.5
  }
]
```


3.6.5 Test scenarios

The module has been tested in isolation and in a full integrated way computing recommendations based on the output of the User Modelling module. The pipeline is currently active and continuously computing recommendations for users as new events occur (e.g. documents are added, user profiles are updated, etc).

3.6.6 Installation and administration guidelines

The Recommender module is composed of several submodules all deployable by using docker orchestration technologies (e.g Cattle, Kubernets, etc.). It is easily deployable using CPN platform's orchestration dashboard (based on Rancher) and a docker-compose file with all the relevant settings has been provided.

Co-funded by the Horizon 2020
Framework Programme of the European Union

4 CONCLUSIONS

This Deliverable reports on the implementation of the technological infrastructure of the 1st prototype of the CPN platform. Subsequent versions of the platform components are expected to provide updated versions of the currently available components, APIs, and services, along with possible new components, APIs, and services, in order to adapt to possible new requirements and functionality needed by the constantly evolving CPN platform.

The components, APIs, and services included in the first version of the platform customization infrastructure and components described in this deliverable have been designed and developed according to the user requirements, as described in deliverable D1.1 “User Requirements Model”.

Co-funded by the Horizon 2020
Framework Programme of the European Union

5 REFERENCES

[1] CPN: D1.1 User Requirements Model

[2] CPN: D2.1 CPN Reference Architecture

[3] CPN: D3.1 Initial Design & APIs of Technology Bricks

Co-funded by the Horizon 2020
Framework Programme of the European Union