Grant Agreement No.: 823783
Call: H2020-FETPROACT-2018-2020

Topic: H2020-FETPROACT-2018-01
Type of action: RIA

D6.3 WENET PLATFORM v.2.0

Revision: v.1.0

Work package	WP 6
Task	Task 6.2, 6.3, 6.4
Due date	30/06/2021
Submission date	25/06/2021
Deliverable lead	UH
Version	1.0
Authors	Bruno Rosell i Gui (CSIC) Isidoros Perikos (OUC) Christos Michael (OUC) William Droz (IDIAP) Avi Segal (BGU)

	Daniel Ben Zaken (BGU) Marcelo Rodas Britez (UNITN) Peter Kun (AAU) Amaia De Götzen (AAU) Rossana Bartolacelli (UH) Nicolò Pomini (UH) Stefano Tavonatti (UH Samuel Bortolin (UH) Carlo Caprini (UH) Daniele Miorandi (UH)
Reviewers	Federico M. Facca (MARTEL)

Abstract	This deliverable covers the second release of the WeNet platform and applications. It contains a description of the software components developed, their specifications, implementation and deployment instructions. All platform components presented are released publicly under an open source license. The WeNet platform and the Ask4Help app have been successfully used within the first round of project pilots in March-Junel 2021.
Keywords	platform, app, software

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	19/02/2021	Structure of the document	Daniele Miorandi (UH)
V0.2	23/03/2021	Revised content after initial checkpoint with the internal reviewer	Daniele Miorandi (UH)
v0.3	5/5/2021	Revised content	Bruno Rosell i Gui (CSIC), William Droz (IDIAP) , Daniele Miorandi UH)
v0.4	10/5/2021	Revised and updated content	Carlo Caprini (UH), Marcelo Rodas Britez (UNITN), Avi Segal (BGU), Daniel Ben Zaken (BGU), Daniele Miorandi (UH)
v0.5	11/05/2021	Revised and cleaned to be sent to internal quality assessment	Carlo Caprini (UH), Daniele Miorandi (UH), Christos Michael (OUC), Marcelo Rodas Britez (UNITN), Bruno Rosell i Gui (CSIC), Peter Kun (AAU), Amalia de Götzen (AAU)
v1.0	25/6/2021	Revised after the internal quality assessment	Daniele Miorandi (UH), Carlo Caprini (UH), Marcelo Rodas Britez (UNITN), Bruno Rosell i Gui (CSIC), William Droz (IDIAP), Christos Michael

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

			(OUC), Stefano Tavonatti (UH), Daniel Ben Zaken (BGU)

DISCLAIMER

COPYRIGHT NOTICE

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

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

EXECUTIVE SUMMARY

This deliverable describes:

- the second release of the WeNet platform;
- the first release of the Ask4Help chat app, which was used in pilots in Italy, Denmark, Mongolia, Paraguay and the United Kingdom.

All software implemented is described in terms of functionality and interfaces exposed. Links to source code and docker images are provided. Licensing information is also included.

The material contained in this report is considered sufficient for an DevOps-skilled person to deploy a complete instance of the WeNet platform and of the Ask4Help app. Formally, according to the DoA, the report is confidential and meant for internal use within the Consortium, in particular for configuring and running pilot deployments. Yet, the material contained in this deliverable will be also used as reference technical documentation for supporting the project Open Call.

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

TABLE OF CONTENTS

Disclaimer	3
Copyright notice	3
EXECUTIVE SUMMARY	4
TABLE OF CONTENTS	5
LIST OF FIGURES	8
LIST OF TABLE	8
ABBREVIATIONS	9
1. INTRODUCTION	10
2. THE WENET PLATFORM	12
2.1 Structure and functionality	12
2.2 Components	15
2.2.1 Service APIs	15
2.2.1.1 Functionality	15
2.2.1.2 Interfaces	15
2.2.1.3 Source code	15
2.2.1.4 Deployment instructions	15
2.2.1.5 Changes from previous version	16
2.2.2 Commons APIs and ILogBase	16
2.2.2.1 Functionality	16
2.2.2.2 Interfaces	16
2.2.2.3 Source code	17
2.2.2.4 Deployment instructions	17
2.2.2.5 Changes from previous version	17
2.2.3 Authentication Server	17
2.2.3.1 Functionality	17
2.2.3.2 Interfaces	18
2.2.3.3 Source code	18
2.2.3.4 Deployment instructions	18
2.2.3.5 Changes from previous version	18
2.2.4 Task Manager	19
2.2.4.1 Functionality	19
2.2.4.2 Interfaces	20
2.2.4.3 Source code	20
2.2.4.4 Deployment instructions	20

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

2.2.4.5 Changes from previous version 20
2.2.5 Interactions Logger and Analytics 21
2.2.5.1 Functionality 21
2.2.5.2 Interfaces 21
2.2.5.3 Source code 21
2.2.5.4 Deployment Instructions 21
2.2.5.5 Changes from previous version 22
2.2.6 Interaction Protocol Engine 22
2.2.6.1 Functionality 22
2.2.6.2 Interfaces 23
2.2.6.3 Source code 23
2.2.6.4 Deployment instructions 23
2.2.6.5 Changes from previous version 23
2.2.7 Incentive Server 23
2.2.7.1 Functionality 23
2.2.7.2 Interfaces 24
2.2.7.3 Source code 24
2.2.7.4 Deployment instructions 24
2.2.7.5 Changes from previous version 24
2.2.8 Profile Manager 25
2.2.8.1 Functionality 25
2.2.8.2 Interfaces 25
2.2.8.3 Source code 25
2.2.8.4 Deployment instructions 25
2.2.8.5 Changes from previous version 26
2.2.9 Personal Context Builder 26
2.2.9.1 Functionality 26
2.2.9.2 Interfaces 26
2.2.9.3 Source code 26
2.2.9.4 Deployment instructions 27
2.2.9.5 Changes from previous version 27
2.2.10 Social Context Builder 27
2.2.10.1 Functionality 27
2.2.10.2 Interfaces 28
2.2.10.3 Source code 28
2.2.10.4 Deployment instructions 28

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

2.2.10.5 Changes from previous version 28
2.2.11 WeNet HUB 29
2.2.11.1 Functionality 29
2.2.11.2 Interfaces 29
2.2.11.3 Source code 29
2.2.11.4 Deployment instructions 29
2.2.11.5 Changes from previous version 30
3. THE ASK4HELP APP 30
3.1 Structure and functionality 30
3.1.2 The chatbot 30
3.1.2 The message endpoint 33
3.2 Usage 33
3.3 Source code 34
4. CONCLUSIONS & NEXT STEPS 34
REFERENCES 35

LIST OF FIGURES

Figure 1: The high-level architecture of the WeNet platform (logical view) 13
Figure 2: Ask4Help app - complete chatbot design... 31

LIST OF TABLE

n.a.

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

ABBREVIATIONS

AI	Artificial Intelligence
API	Application programming interface
CDN	Content delivery network
DoA	Description of the Action - Annex 1 of the WeNet Grant Agreement
GDPR	General Data Protection Regulation (EU - 2016/679)
IPE	Interaction protocol engine
IS	Incentive server
IT	Information Technology
ML	Machine Learning
PM	Profile Manager
RI	Research Infrastructure
RPS	Request per second
WP	Work-package

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

1. INTRODUCTION

The main goal of WeNet is to grow and advance the culture, science and engineering methodologies, algorithms and social interaction protocols for an online platform aiming at empowering machine-mediated diversity-aware interactions between people. WeNet takes particular care to ensure that the technology that will be designed and developed will respect users' differences by embodying fundamental features of transparency, fairness, and accountability.

Within the overall project structure, WP6 is tasked with (i) the design and implementation of the WeNet platform (ii) the prototyping of applications to be used in the pilots (iii) the instantiation of the WeNet research infrastructure (iv) the deployment and maintenance of instances of the WeNet platform, applications and research infrastructure to be used in the project pilot experiments.

The WeNet platform [1] is a software framework that allows application developers, innovators and Web entrepreneurs to quickly and easily develop and deploy diversity-aware applications. By 'diversity-aware applications' we identify applications that both recognise and respect diversity in their underpinning data processing operations (in particular in terms of data protection and ethical usage of AI techniques) and that leverage diversity in their user base to provide valued-added services to individuals and communities.

On top of the platform, the Consortium is developing a set of applications, meant to support the target users (university and higher education students) in carrying out activities related to their university life in a more efficient, effective and enjoyable fashion. A number of relevant use cases for said apps have been identified by WP7, where they have also been ranked according to their fit with the specific pilot sites context and the added value brought forward by diversity-awareness. Since the release of D6.2 [3] at M18, also due to the impact of the covid-19 pandemic on campus life, the focus has shifted on the 'ask for help' use case [2], which has been implemented and piloted, thereby complementing the 'eating together' use case implemented in year-2 and presented in D6.2.

The WeNet research infrastructure is a data infrastructure that allows WeNet experimenters[1] to manage the full data lifecycle, including collection and processing (with specific focus on GDPR compliance and ethics), as well as sharing with relevant stakeholders. At the same time, it allows researchers (in particular from social sciences, but not only) to access a large number of experimental datasets relevant for understanding the role of diversity in machine-mediated human interactions.

This deliverable includes (as per DoA) the "Enhanced version of the platform (..), including the documentation. Includes enhanced *version of the Smart University app.*" The deliverable is of type 'Demonstrator', and it is formally, according to the DoA, marked as Confidential. Yet, the material within the deliverable will be made available to Open Call applicants as key reference documentation; accordingly, all the codebase repositories are also getting opened. This move by the Consortium is meant to accelerate the impact maximisation strategy outlined by the Consortium within WP8.

This deliverable includes a description of the software components implemented, and instructions for deploying them.

[1] And third parties, in the longer run.

The work presented in this deliverable builds upon D6.1 [1], which includes a description of the overall WeNet platform architecture and components.

The remainder of the deliverable is organised as follows:

- In Sec. 2 we first briefly revise the structure and functionality of the WeNet platform. We then provide, for each component, the functionality implemented, the available set of APIs, the link to the source code repository (including documentation), the licensing scheme used, instructions for deploying the component and a short summary of changes from the previous version included in D6.2 [3].
- The Ask4Help application is described in Sec. 3. We present the design of the app in terms of dialogue structure and implementation aspects, together with a link to the source code repository.
- Sec. 4 concludes the deliverable with a list of enhancements and new features currently being undertaken, and which will be released over the next few months.

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

2. THE WENET PLATFORM

2.1 STRUCTURE AND FUNCTIONALITY

The WeNet platform consists of *a set of modular, interoperable, open software components providing the required functionality to enable the fast development and deployment of diversity-aware applications.* The WeNet platform will therefore support a plurality of applications (be them mobile, web or chat apps), which will leverage the functionality exposed by the platform. Applications can interact with the platform through a set of purposeful APIs. The platform is able to connect to third-party services and applications for collecting relevant data on user behaviour, including, but not limited to, mobile phone sensors (through a mobile app), calendar applications, social media streams, open data portals etc. Data streams from said third-party services and applications are used - by means of advanced AI/ML methods and algorithms - to personalise the user experience and to properly account for diversity dimensions.

As explained in detail in D6.1 [1] and D6.2 [3], the platform design was driven by the pilot sites requirements, and informed by interactions with the science-oriented WeNet WPs (WP2/3/4/5) as well as with discussion with WP9 on ethical aspects (in terms of both design and operation of the platform). The overall logical architecture of the WeNet platform is described in the following figure. Below we add a brief description of the role and functionality of the various components.

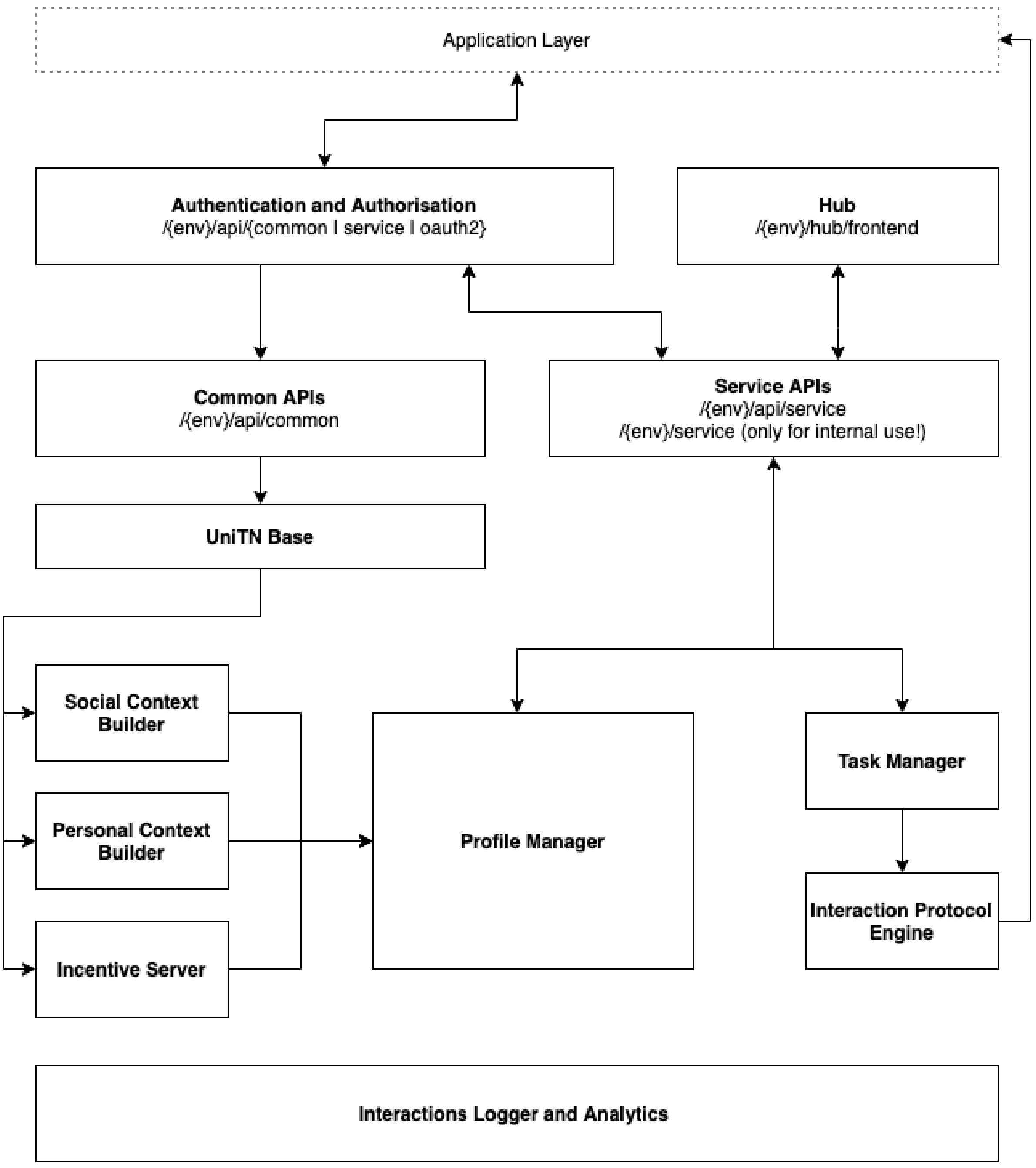


Figure 1: The high-level architecture of the WeNet platform (logical view)

The WeNet **Service APIs** are the ones application developers can leverage to build diversity-aware apps. In particular, the Service APIs allow managing Applications, Users, Profiles, Messages and Tasks. Using them, a developer can register her own app on the platform, receive the credentials for authenticating the requests and univocally associate them to her own app. Through them, users can create an account and login using a set of existing credentials. To this aim, a specific web frontend (the WeNet Hub) has been developed. Through the Service APIs, an application can access a user profile (provided that the user approved such a request). Leveraging them, users can exchange messages, create tasks and/or contribute to them.

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

The **Commons APIs** are used to collect, through dedicated connectors, data from third-party apps. Such data can be used by various AI/ML components to better characterise the diversity dimensions of the users, enriching the user profile and allowing apps to provide a personalised user experience. The WeNet Commons APIs can be used to connect, for example, to a third-party mobile application feeding measurements on the current location of the user, but also to Twitter or Google Calendar, allowing access to tweets and appointments, respectively.

The **Task Manager** is the component responsible for managing tasks and maintaining their status. A task collects, e.g., all the information describing a help request posted by a user on the platform and tracks the various steps necessary for the requester to evaluate volunteers, choosing one to complete his request and verifying the successful completion of the task.

The **Interactions Logger and Analytics** plays a twofold role. First, it collects the messages exchanged within the users' conversations as well as the platform components' log messages. Leveraging its message logging functionality, this component has a key role in the platform: it allows to analyse and audit past conversations, in particular for understanding the impact of the behaviour and configuration of the different platform components.
Second, this component provides analytics both at message and component level. Such information can be used to monitor the intensity and content of the WeNet-mediated social interactions, as well as to monitor and manage the status of the platform components.

The **Authentication and Authorisation Server** is responsible for authenticating and authorizing requests from applications, users and platform components.

The **Interaction Protocol Engine** is the component responsible for managing the interaction between users, in particular for guaranteeing the respect of individual, community and task-level norms.

The **Incentive Server** is responsible for deploying all the incentive measures required for triggering the user interest in contributing to tasks and for maintaining users engaged over time.

The **Profile Manager** is the component responsible for storing and maintaining the WeNet user profiles, while ensuring that access to the information by applications is strictly controlled.

The **Personal Context Builder** is responsible for analysing the data collected by the various Common APIs connectors and identifying relevant routines describing the users' habits, through the application of purposeful ML methods.

The **Social Context Builder** is responsible for analysing the data collected by the various Common APIs connectors and for learning information describing the social preferences, relations and interactions of users.

The **iLogBase** manages in a scalable way data coming from the WeNet Commons APIs and serves said data to other inner platform components.

2.2 Components

In the following sections we describe the current implementation of each WeNet platform component, as listed in the previous section. For each component, we describe the functionality implemented, the APIs exposed, we provide a link to the source code and a link to the relevant docker image(s).

2.2.1 Service APIs

2.2.1.1 Functionality

The Service APIs component allows applications to interact with the WeNet platform.

It exposes dedicated endpoints allowing to manage:

- applications;
- users;
- user profiles;
- tasks and the associated transactions.

The APIs defined in this component are meant to be used mainly by WeNet applications (either developed by WeNet consortium or by third parties).

2.2.1.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-service_api-openapi.yaml.

2.2.1.3 Source code

The source code of this component is hosted on BitBucket and is available here: https://bitbucket.org/wenet/wenet-service-api/src/master/. The Readme file includes the description and the step-by-step instructions for correctly running the Service APIs. Also, the steps necessary for building a Docker image of the component are described. The code of this component is released under an Apache 2.0 license.

2.2.1.4 Deployment instructions

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

The repository includes a dedicated script for creating a Docker image of the Service APIs and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service.

The Docker images associated with the various releases of the Service APIs are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.1.5 Changes from previous version

Since the previous version, the following main changes have been applied.

- The component has been completely detached from the WeNet Hub Component. They were initially sharing the reading/writing access to the same database. This has not been resolved and the Service APIs component leverages the Hub APIs in order to access the required information.
- The integration with the Interaction Logger and Analytics has been completed. It is now possible for third party applications to register the history of exchanged messages.

Support for unit testing and test coverage is present in the repository. The development of the platform components has been conducted privately until now so there are no processes in places for supporting and managing pull requests. This is something we will be focusing on in the next months.

2.2.2 Commons APIs and ILogBase

2.2.2.1 Functionality

The Commons APIs allow developers to connect external data streams to the WeNet platform. These can be used to analyse and understand user behaviour and context. Within the framework of the project, the streams of data are mainly coming from the iLog application (user sensors and user feedback), but it is possible to use them to connect other external services (e.g., social media, user calendar etc.). The iLogBase component is responsible for managing streams of data from third-party apps. The services of this component allow saving and sharing the streams of data so that other WeNet components can access them.

The access for internal or external components is handled with authentication procedures for registering the users of the components with different levels of authorizations.

These components provide services for handling big data. The services include the management of streams of data with batch services and with subscription/notification services. The internal components of the WeNet infrastructure can access batch data or register to a feed subscription.

The WeNet Commons APIs and iLogBase components are tightly coupled and shipped together as a single software package.

2.2.2.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here:

- http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-common_api-openapi.yaml
- http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-ilogbase-openapi.yaml

2.2.2.3 SOURCE CODE

The source code of this component is hosted on BitBucket. The code is released under an Apache 2.0 license.

2.2.2.4 DEPLOYMENT INSTRUCTIONS

The repository includes a dedicated script for creating a Docker image of the iLogBase and Commons APIs components and thus running the components directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. There are 3 dependencies related to these services: 1) MySQL instance; 2) Cassandra instance; and 3) docker-batch-subscription-data (UNITN module).

The Docker images associated with the various releases of iLogBase and Commons APIs are hosted and publicly available on the Docker Hub in the following link: https://hub.docker.com/r/internetofus/ilogbase. Deployment instructions are included in the docker image repository documentation.

2.2.2.5 CHANGES FROM PREVIOUS VERSION

The main changes done from the previous release are the following:

1. New streams of data to handle Social Information useful for the Social Context Builder have been added. The new streams are the following: Social Events, Social Profile, Social Posts and Social Relations.
2. A new model for data representation has been implemented. The new structure includes, for each data stream:
    a. **ts**: timestamp;
    b. **payload**: is the data of the stream;
    c. **meta**: additional information for identifying the data stream.
3. We add the **appId** field for putting the application that is handling the data.
4. We move the **experimentId** as part of the **meta** field for better organization of the information.

2.2.3 Authentication Server

2.2.3.1 FUNCTIONALITY

The Authentication Server provides the following main functionality:

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

1. enabling request authentication in the communication between the various components of the WeNet platform;
2. enabling request authentication for external applications build on top of the WeNet platform;
3. enforcing user authentication;

This component is currently mapping the urls of the components platform, deployed on the infrastructure of the partners responsible for them, to default and static urls.

2.2.3.2 INTERFACES

There is no explicit interface dedicated to the Authentication server. In fact, this component behaves as a proxy in all http requests; therefore it is always included. It forwards authenticated and authorised requests to the correct platform components.

2.2.3.3 SOURCE CODE

The Authentication Server relies on the open source API management tool Kong (https://konghq.com/).

2.2.3.4 DEPLOYMENT INSTRUCTIONS

The deployment of the Authentication Server is tracked in the platform deployment dedicated repository and handled by the Ansible playbook responsible for the deployment of the complete WeNet platform. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. Kong needs a database to store its configuration and to manage the authentication, so a Postgres database has been added to the deployment. Kong has a series of migrations that should be applied to the database, to do so two other components have been added to the docker-compose: kong-migrations and kong-migrations-up.

Optionally It is possible to add a dashboard for monitoring and managing kong. In the docker-compose is available the konga service which contains an instance of the opensource Konga dashboard. This dashboard requires a database, so a dedicated Postgres database has been added to docker-compose. Also, the konga-prepare has been added to perform all the migrations

2.2.3.5 CHANGES FROM PREVIOUS VERSION

The Authentication server now provides authentication and authorization for both platform internal components and external third-party applications.

In particular, cross-component authentication is achieved by means of component-dedicated access tokens. Each component has been associated with an access token that is to be specified in all internal requests. The Authentication component is responsible for verifying each token authenticity: only authenticated requests are then forwarded to the target component. This makes it possible for the platform components not to require any type of request authentication.

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

Third-party applications' authentication and authorisation can be completed with two strategies. On the one hand, application-specific tokens can be used in order to perform general purpose requests. On the other hand, OAuth2 tokens are required in order to complete any user-specific requests.

2.2.4 Task Manager

2.2.4.1 Functionality

The task manager component is the one responsible for storing and maintaining the task and task types, and managing the actions that can modify the task state.

A task is an instance of a task type. The task type contains the description of the attributes necessary to define the task, the list of possible transactions (actions) that can be done for a task of given type, and a set of norms that define how the task can change its state. For example, considering a task type representing a user asking for help to the others members of a community, we would have:

- ***Attributes***
    - **question: the question that the user asks.**
    - **numUsers**: as the number of users to ask the question
- ***Transactions***
    - **reply**: when a user wants to provide an answer to the question
    - **selectAnswer:** when a user selects an answer
    - **moreAnswers:** when the user wants to get more answers from other users that were not selected before.
    - **cancel:** when the user that asked the questions deletes the question.
- ***Norms***
    - When a task is created, notify a subset of friends about the question and ask them to answer.
    - When a user provides an answer, notify the task requester and append the newly received answer to the list of all answers received.
    - When the requester selects an answer, then notify the user that its answer is the best, notify the other users that they help is no longer needed and mark the task as closed.
    - When the requester asks for more answers, create a list of users that have not been notified yet and notify them. If the subset is empty, notify the requester that it can not receive more answers.
    - When requester cancels the task, inform to all the users who were asked to provide an answer that they help is n longer needed and mark the task as closed

In this example, we could have the following attributes:

- **question**: Which is the best Pizza?
- **numUsers**: 5
- **state**: Open
- **friends**: User2, User89, user78, user780, user8
- **answers: [{user:**: User78,answer:'The margarita is the best'}, **{user:** User2,answer:'The tropical is the best'}]

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

The transactions represent asynchronous actions that can change the task state. When a user, an application or other WeNet components want to change the state of a task, they have to post a transaction to the task manager. The task manager checks that the transaction is correct according to the task type; then the transaction is sent to the interaction protocol engine to verify the task, community and user norms. In other words, the changes of the state are managed through norms that are evaluated on the interaction protocol engine, and not by the task manager directly.

2.2.4.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-task_manager-openapi.yaml .

2.2.4.3 Source code

The source code of this component is hosted on BitBucket and is available here: https://bitbucket.org/wenet/wenet-task-manager/src/master/. The README.md file includes the description and the step-by-step instructions for correctly running the task manager. Also, the steps necessary for building a Docker image of the component are described. The code of this component is released under the Apache V2 License.

2.2.4.4 Deployment instructions

The repository includes a dedicated script for creating a Docker image of the Task Manager and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. This component requires a MongoDB database.

The Docker images associated with the various releases of the Task Manager are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.4.5 Changes from previous version

The main changes done over this component from the previous deliverable are:

- Published into dockerhub to allow any person to access it.
- Integrated the authentication key to interact with the other components on the platform.
- The task model has been modified to contain the information of the transactions that have applied to it. It now also includes the callback messages that have been sent to the users on the execution of tasks.
- Auto-register the task types necessary to run the social eating and Ask4Help pilots. Also provide a version of these types using the new protocol norms defined into the interaction protocol engine component.

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

2.2.5 Interactions Logger and Analytics

2.2.5.1 FUNCTIONALITY

The Interaction Logger and Analytics component is responsible for collecting and storing all the messages, and associated metadata, that are exchanged during the conversations that take place during the organization and management of a task together with the low software logging events. In addition, this component is also responsible for computing key analytics describing the KPIs of each conversational application built on top of the WeNet platform.

This component is built upon a logger and analytics platform developed by U-Hopper in the context of Memorable Experiences project (carried out within the ERDF Regional Operational Programme 2014-2020 of the Autonomous Province of Trento and co-financed by the European Union through the European Fund for Regional Development, for the Italian state and the Province of Trento).

Both Logger and Analytics data contain key information allowing to monitor the success of a conversational application. On the one hand, message data is going to be of great help to the applications' developers who will have the possibility to understand how conversations are evolving from a technical point of view, easily identifying potential issues and blocking factors. On the other hand, analytics data is going to offer key information on the performance and effectiveness of a service.

The Message model has been defined in order to provide a general structure that allows log messages exchanged not only on the most common messaging platforms (e.g. Facebook, Telegram) but also on custom conversational services. The Log model has been structured in order to include the most common features of a code logging event. Among the others: component name, severity, timestamp, message and a general metadata field.

2.2.5.2 INTERFACES

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-logging_analytics-openapi.yaml.

2.2.5.3 SOURCE CODE

The source code of this component is tracked on BitBucket. The code is released under an Apache 2.0 license. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. This component takes advantage of an Elasticsearch database for storing messages, logs and computed analytics.

2.2.5.4 DEPLOYMENT INSTRUCTIONS

The repository includes a dedicated script for creating a Docker image of the Analytics and Logging component and thus running the components directly within Docker.

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

The Docker images associated with the various releases of Analytics and Logging are hosted and publicly available on the Docker Hub in the following link: https://hub.docker.com/r/internetofus/logger.

2.2.5.5 CHANGES FROM PREVIOUS VERSION

In this second version of the platform, the Analytics and Logging component has been integrated in the platform. Message logging has been integrated and enabled during the pilots run at M26.

Component logging has not been integrated yet. This will be the focus for the next platform release.

2.2.6 Interaction Protocol Engine

2.2.6.1 FUNCTIONALITY

The interaction protocol engine component is the one responsible for guaranteeing that interactions between WeNet users follow the rules encoded in the relevant norms. The interaction between users is modelled as an exchange of messages.

When a user sends a message through the service API, the message is sent to the norm interpreter that will interpret the norms from the point of view of the sender user. This interpreter needs to first verify that the message does not violate any of the relevant norms, which includes the community norms, the task norms, the sender's individual norms, as well as the context-dependent norms that are attached to this message. If the message violates any of those norms, an error message is sent back to the user. However, if the message obeys the norms, then the norm interpreter needs to decide what to do next, usually translating the request into a set of new messages to deliver to other peers. This decision follows from the community, individual and context-dependent norms, and takes the user's profile (both public and private) into account as needed. If the message is sent to the interpreter of another user, the norm interpreter of the recipient needs to first verify that the message does not violate any of the relevant norms. This re-checking upon receipt ensures that the sender's norm engine has not been manipulated. If the message violates any of the community norms, then it may either be discarded, or if the community norms require sanctioning, then the appropriate sanctions should be executed. However, if the action obeys the community norms, then the norm interpreter needs to decide what to do next, which is usually translated into sending messages to other peers and/or sending messages to its user. This decision takes into consideration the community norms, the context-dependent norms that are attached to the message, the individual private norms of the human whose interpreter has received this message, as well as their local profile (both private and public). This ensures that the interpreter abides by human's private norms without leaking any of their private norms and profile.

There are norms on the individual (user level), the task level, and the community level. An individual's norm might be "Suppress incoming messages at night" (and this will be applied for the user who sets this norm only). A task norm might be "Don't ask my ex partner" (so that would be applied for a specific task only). A community norm might be "If you don't volunteer, you cannot ask for help" and it would be enforced on everyone. Given the above, this means that norms will be attached to users, tasks and communities.

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

2.2.6.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-interaction_protocol_engine-openapi.yaml .

2.2.6.3 Source code

The source code of this component is tracked on BitBucket and is available here: https://bitbucket.org/wenet/wenet-interaction-protocol-engine/src/master/. The README.md file includes the description and the step-by-step instructions for correctly running the interaction protocol engine. Also, the steps necessary for building a Docker image of the component are described.

The code of this component is released under the Apache V2 License.

2.2.6.4 Deployment instructions

The repository includes a dedicated script for creating a Docker image of the Interaction Protocol Engine and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. This component requires a MongoDB database.

The Docker images associated with the various releases of the Interaction Protocol Engine are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.6.5 Changes from previous version

The main changes done over this component from the previous deliverable are:

- Published into docker hub to allow any person to access it.
- Integrated the authentication key to interact with the other components on the platform.
- Integrated the norm engine written in SWI Prolog and used to validate the norms.

2.2.7 Incentive Server

2.2.7.1 Functionality

The Incentive Server is responsible for generating diversity-aware incentives used during WeNet's individual and group interactions to maximize the probability of successful interactions, i.e., interactions that are beneficial to accomplishing the goals of the different participants in the WeNet system. The component leverages the information provided by the other components for identifying the appropriate incentives to users while adhering to WeNet norms. The current APIs support message-based and badge-based incentives. An additional norm endpoint is supplied, to be used for internal testing and validation of the current incentive

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

server ML mechanisms; these endpoints may be deprecated and superseded in future releases.

The server exposes dedicated endpoints allowing to manage:

- badge incentives;
- message incentives;
- norms relating to the incentive server behaviour.

Badges incentives are based on the Open Badges 2.0 (OBv2) API specification.

The APIs defined in this component are meant to be used by other components of the WeNet platform (but the norms endpoint, which is for internal testing only).

2.2.7.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here.

2.2.7.3 Source code

The source code of this component is tracked on BitBucket and is available here. The repository contains a postman collection which includes the description and the step-by-step instructions for correctly running the Incentive server. Also, the steps necessary for building a Docker image of the component are described.
The code of this component is released under an Apache 2.0 license.

2.2.7.4 Deployment instructions

The repository includes a dedicated script for creating a Docker image of the Incentive Server and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service.

This component depends on the WeNet task manager. The docker-compose also include three other dependencies:

- concentric sky,badgr-server (open-source)
- MySql server
- cron service

The Docker images associated with the various releases of the Incentive Server are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.7.5 Changes from previous version

The main changes done in the Incentive Server component from the previous deliverable are the following.

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

- Published component into dockerhub to allow usage by third parties.
- Integrated new authentication mechanism to enable interaction with other components on the platform based on the newly defined authentication mechanism.
- Deployed component's web services in production using Nginx and WSGI.
- Integrated with the task manager, profile manager and additional components which were supplied by other work packages as part of the overall WeNet project.
- Defined, customized and supported incentive messages and incentive badges as part of the Ask4Help WeNet app for multiple communities.
- Defined and executing an experiment.
- Defined and executed an A/B testing experiment in the Ask4Help pilot.

2.2.8 Profile Manager

2.2.8.1 FUNCTIONALITY

The profile manager component is responsible for storing and maintaining the WeNet user profiles and their history. A user profile consists of a set of attributes that define the state of the user. Some of these attributes are filled in by the user at registration time (e.g. name, email, telephone) or at runtime by other components of the platform (e.g. social relationships, routines). Every time a profile is modified, a copy of the previous profile values is stored in the history. This allows platform components to analyse how the user profile evolves over time.

Another responsibility of the profile manager is to evaluate the trust of one user in another one when the latter is performing some action. The trust is dynamic and will be updated every time they collaborate to achieve a task. When a user has received help it can rate the performance of the user that has helped it. For this it has to post a performance rating event to the profile manager. These events are used by the profile manager when it has to provide the trust that has an user that another does a certain action.

2.2.8.2 INTERFACES

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-profile_manager-openapi.yaml .

2.2.8.3 SOURCE CODE

The source code of this component is tracked on BitBucket and is available here: https://bitbucket.org/wenet/wenet-profile-manager/src/master/. The README.md file includes the description and the step-by-step instructions for correctly running the profile manager. Also, the steps necessary for building a Docker image of the component are described. The code of this component is released under the Apache V2 License.

2.2.8.4 DEPLOYMENT INSTRUCTIONS

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

The repository includes a dedicated script for creating a Docker image of the Profile Manager and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. This component requires a MongoDB database.

The Docker images associated with the various releases of the Profile Manager are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.8.5 Changes from previous version

The main changes done over this component from the previous deliverable are:

- Published into dockerhub.
- Integrated the authentication key to interact with the other components on the platform.

2.2.9 Personal Context Builder

2.2.9.1 Functionality

The Personal Context Builder is responsible for building and maintaining users' routines. In WeNet, a routine allows to describe user habits and is dynamically updated by analysing the user data streams that are collected via the southbound Common REST APIs.

In particular, two different routine representations will be made available.

- The Semantic routine is a human-readable routine representation that can be easily leveraged by the other WeNet components.

- The Embedded routine consists in the raw routine representation that matches the output of the machine learning algorithms applied for computing them.

The information generated on routines is pushed to the profile manager and maintains the profile constantly updated.

We also provide real-time APIs - to access the current user context - in a dedicated subcomponent.

2.2.9.2 Interfaces

Detailed OpenAPI documentation can be found here : http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-personal_context_builder-openapi.json

For the real-time subcomponent : https://lab.idiap.ch/devel/hub/wenet/docs

2.2.9.3 Source code

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

License: Apache-2.0

https://bitbucket.org/wenet/wenet-personal-context-builder/src/master/

2.2.9.4 DEPLOYMENT INSTRUCTIONS

The two components have ready-to-use docker images:

- real-time : https://hub.docker.com/repository/docker/internetofus/wenet-realtime
- personal_context_builder : https://hub.docker.com/repository/docker/internetofus/personal-context-builder

The main system that exposes the APIs can be deployed by using docker-compose. There is one docker-compose file for the dev environment that uses the latest build and another for the production environment with a pinned version.

To get all functionalities of the personal_context_builder, you need two redis databases to store the embedded routines and models.

For more detailed information, you can consult the README of the project.

For the real-time subcomponent, there is an example on how to deploy it on kubernetes : https://bitbucket.org/wenet/wenet-realtime/src/master/wenet-realtime.yaml and a docker-compose https://bitbucket.org/wenet/wenet-realtime/src/master/docker-compose.yaml

This also a required part of the main wenet component : https://bitbucket.org/wenet/wenet-personal-context-builder/src/master/personal_context_builder_redacted.yaml

2.2.9.5 CHANGES FROM PREVIOUS VERSION

The real-time service is not a standalone component separated from the main component (personal_context_builder).

The following functionalities have been added to the main component:

- ability to update the real-time subcomponent (like a service)
- activate/disable the data generator from Streambase

We also added support for CI/CD, so all components are published to dockerhub automatically.

2.2.10 Social Context Builder

2.2.10.1 FUNCTIONALITY

The Social Context Builder is responsible for building and maintaining the social details of a user profile. The core functionalities of the components are the following:

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

- Learning of the social relationships linking WeNet users and estimation of their strength, initially computed based on user data from third-party platforms (e.g., social media) and is updated based on the interaction of the users within WeNet.
- Ranking of volunteers (users who accepted a given task request).
- Provide arguments and the reasons on why a given volunteer (identified by WP5) for the requestor's task should eventually be accepted by the requestor to help with the task. Accordingly, the requestor is able to question the explanations that are provided, and coach the component towards learning to offer personalized explanations to the requestor.

2.2.10.2 Interfaces

All interfaces of this component are described in Open API format:

http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-social_context_builder-openapi.json

2.2.10.3 Source code

The source code is available at the Bitbucket repository https://bitbucket.org/wenet/wenet-social-context-builder/src/master/
The repository includes instructions on how to run the component locally with a Docker image.
Additionally there are examples of data requests using the live development server. The code of this component is released under an Apache 2.0 license.

2.2.10.4 Deployment instructions

The repository includes a dedicated script for creating a Docker image of the Social Context Builder and thus running the component directly within Docker. A *docker-compose.yml* example is present in the repository, it can be used to jump start the setup of this service. All the required packages used by the component are being noted in the requirements.txt file and installed automatically in the build. The service requires a SQLite database.

The Docker images associated with the various releases of the Social Context Builder are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.10.5 Changes from previous version

Since the previous version, the following main changes have been applied.

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

- Published publicly to dockerhub.
- Added a database to hold social network ids and WeNet ids relationships
- Added stream endpoints for the new social profile and social relationships data structures.
- Integrated authentication key for interaction with the rest of the platform components.

2.2.11 WeNet HUB

2.2.11.1 Functionality

The WeNet HUB is responsible for allowing WeNet users to manage their profile and applications. It also provides developers with the tools for managing their applications.

In order to use a WeNet application a WeNet user should have been created. This is possible via the WeNet HUB. Any WeNet application will redirect its users to the HUB if a WeNet user can not be identified. Once registered and authenticated, a user can take advantage of the HUB for:

- managing his profile information (this include general demographic details);
- browsing available WeNet applications;
- enabling the WeNet applications of interest in order to start using them;
- managing enabled applications.

The WeNet HUB is also the place where new WeNet applications can be created and configured. Any user can create his own WeNet application by becoming a developer. This can be done by enabling the developer mode in the user account settings. A few profile information will need to be set in order to complete this process. Once a developer, a user has access to a dedicated section of the HUB where it is possible to configure applications from a technical point of view.

2.2.11.2 Interfaces

All interfaces of this component are described in detail in the project documentation in Open API format. The documentation is publicly available here: http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-hub-openapi.yaml.

2.2.11.3 Source code

The source code of this component is tracked on BitBucket and is available here: https://bitbucket.org/wenet/wenet-hub/src/master/. The Readme file includes the description and the step-by-step instructions for correctly running the HUB. The code of this component is released under an Apache 2.0 license.

2.2.11.4 Deployment instructions

The repository includes a dedicated script for creating a Docker image of the WeNet Hub and thus running the component directly within Docker. A *docker-compose.yml* example is present

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

in the repository, it can be used to jump start the setup of this service. This component depends on a MySQL database (for storing managed data) and on a Redis one (for handling web sessions).

The Docker images associated with the various releases of the WeNet Hub are hosted and publicly available on the Docker Hub. Deployment instructions are included in the docker image repository documentation.

2.2.11.5 CHANGES FROM PREVIOUS VERSION

The design and implementation focus has been on improving the developer section of the Hub. Also, the support for applications available and won badges has been introduced.

3. THE ASK4HELP APP

In terms of applications, initially the choice by the Consortium was to implement the social eating use case [2], which led to the implementation of a first prototype, described in D6.2 [3]. Yet, the arrival of the covid-19 pandemics and the consequent changes in social habits and campus life has forced us to rethink the app and to revise the priorities in terms of use cases. Together with WP7, we decided therefore to shift the focus on the “Ask for Help” use case [2]. This led to the design and development of a new app (Ask4Help), meant to allow WeNet users to ask questions to the community, answer open questions and approve the best provided answer for each question. The app uses the resources and functionality exposed by the platform to provide users with a personalised experience based on diversity dimensions. The Ask4Help app was used by pilots in Italy, Denmark, Mongolia, Paraguay and the United Kingdom in the period March-June 2021.

3.1 STRUCTURE AND FUNCTIONALITY

The app was developed using Telegram[2] as an enabling platform. It was therefore designed and implemented as a chat application, powered by a chatbot engine which interacts with the WeNet platform endpoints.

The chatbot has two main components:

1. the chatbot itself, that is a webhook receiving and sending messages from/to the Telegram APIs;
2. an HTTPS endpoint, to let the chatbot receive messages from WeNet.

3.1.2 The chatbot

The overall chatbot design is condensed in the following figure, which represents all conversation steps, with the messages to be used and the actions foreseen.

[2] https://telegram.org/

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

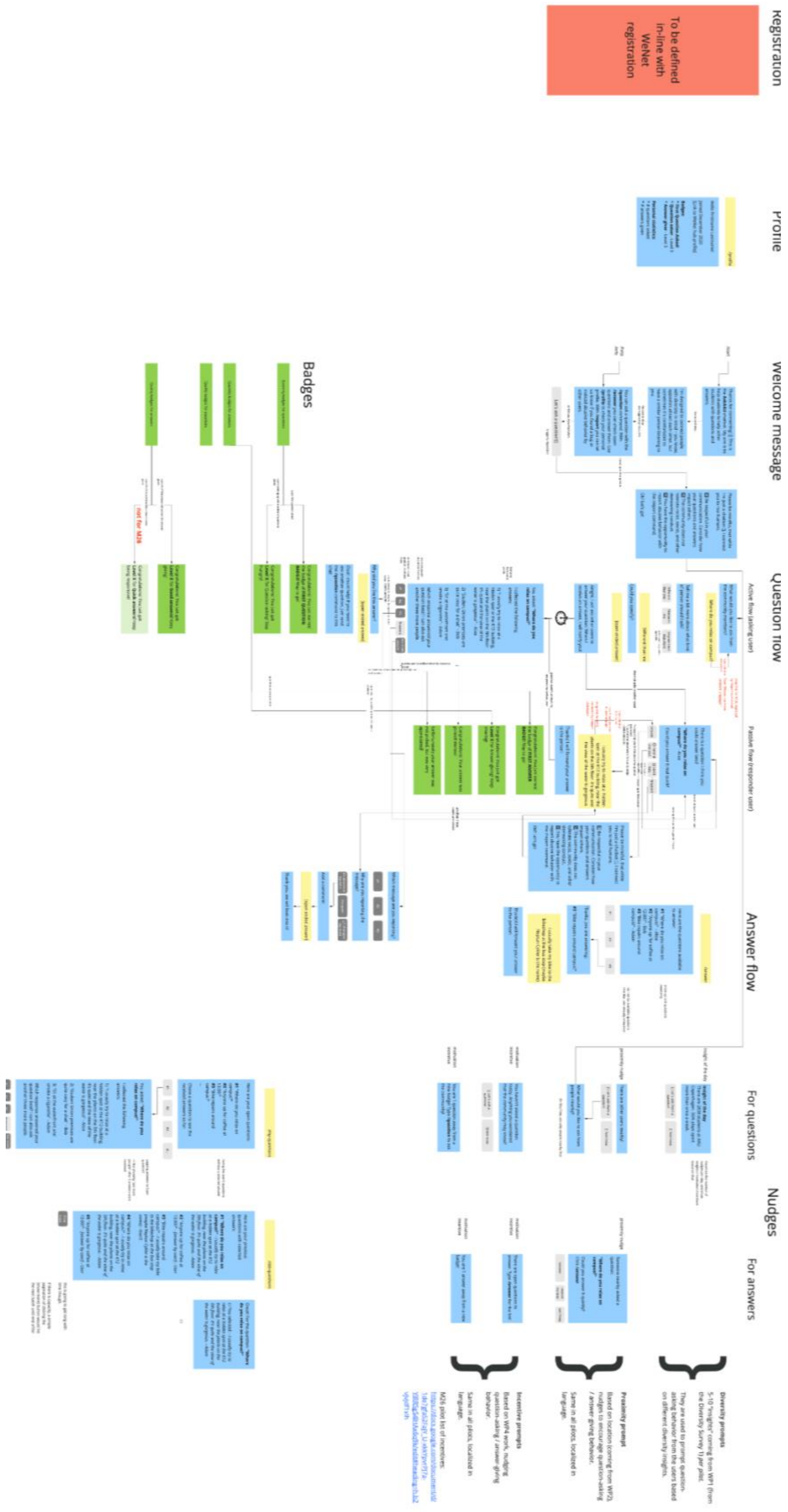
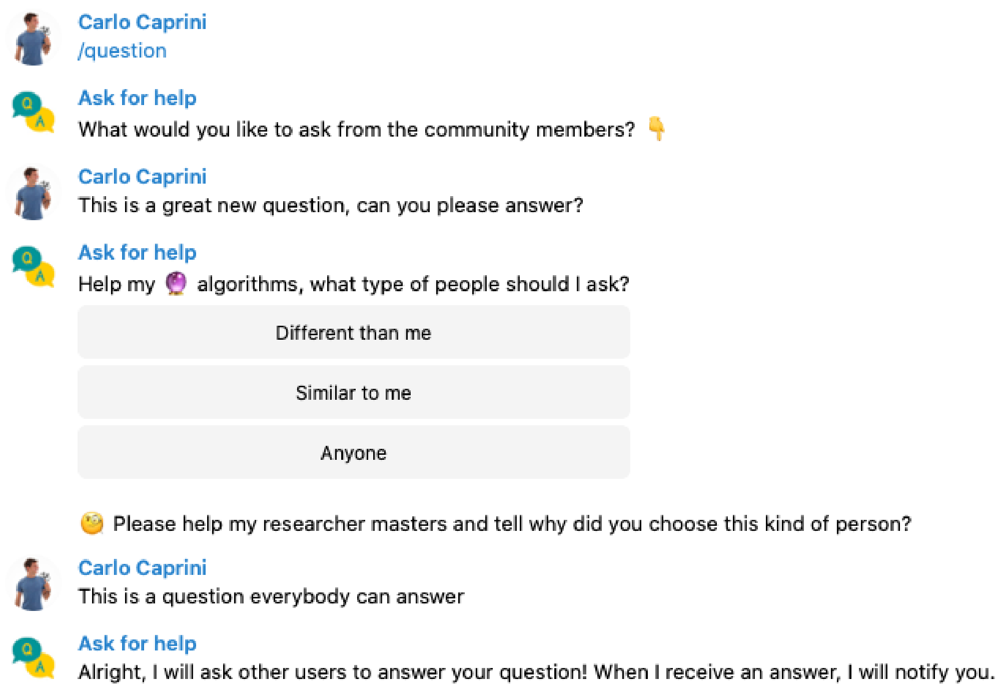


Figure 2: Ask4Help app - complete chatbot design.

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

The chatbot is implemented as a deterministic automata developed in Python, allowing WeNet users to create and manage questions, answer open questions and resolve existing questions by selecting the best provided answer.

The process to create a new question is a 3-step message flow and it is started by typing the command `/question` to the bot chat: the user is asked to type a question, select a target audience the question should be proposed to and motivate the choice.



Carlo Caprini
/question

Ask for help
What would you like to ask from the community members? 👇

Carlo Caprini
This is a great new question, can you please answer?

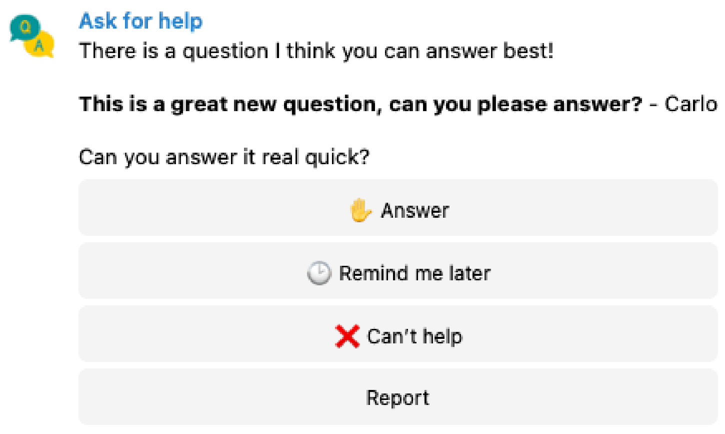
Ask for help
Help my 🔮 algorithms, what type of people should I ask?

🧐 Please help my researcher masters and tell why did you choose this kind of person?

Carlo Caprini
This is a question everybody can answer

Ask for help
Alright, I will ask other users to answer your question! When I receive an answer, I will notify you.

Once the question has been created, it is up to the platform to select the users whose participation in the discussion may be most beneficial and to propose the question. It is up to each user to decide if answering the question, requesting for a later reminder, specify they can not contribute or report the question as inappropriate.



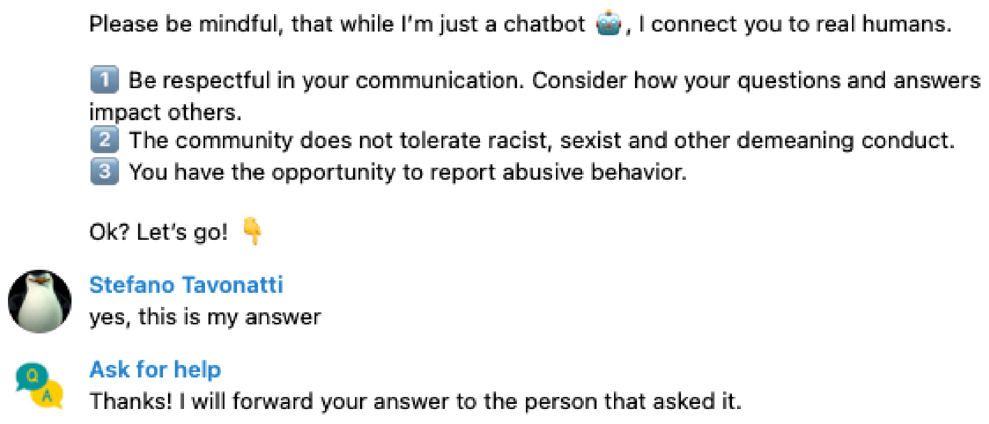
Ask for help
There is a question I think you can answer best!

This is a great new question, can you please answer? - Carlo

Can you answer it real quick?

When collecting a new answer, the bot always reminds the user a few best practices to consider in order to guarantee a good quality in the conversation.

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



Please be mindful, that while I'm just a chatbot 🤖, I connect you to real humans.

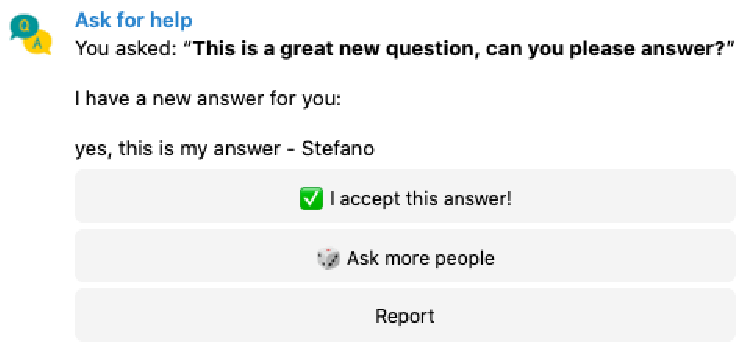
1️⃣ Be respectful in your communication. Consider how your questions and answers impact others.
2️⃣ The community does not tolerate racist, sexist and other demeaning conduct.
3️⃣ You have the opportunity to report abusive behavior.

Ok? Let's go! 👇

Stefano Tavonatti
yes, this is my answer

Ask for help
Thanks! I will forward your answer to the person that asked it.

The creator of a question is currently the only one receiving new answers as soon as they are provided. It is up to the question creator to accept a specific answer or report it as inappropriate. It is also possible to request WeNet to propose the same question to a broader group of users.



Ask for help
You asked: "**This is a great new question, can you please answer?**"

I have a new answer for you:

yes, this is my answer - Stefano

Once an answer is accepted, the associated question is considered resolved.

3.1.2 The message endpoint

An HTTPS endpoint was implemented to allow the WeNet platform to send messages to the bot. Messages are organized in three different groups:

1. **Events**, used when a user logs into WeNet with a Telegram account for the *Let's eat together* task. In this way the bot is able to send a welcome message to the new user.
2. **Notifications**, used for standard messages, such as when a new task suggestion is available, or when a volunteer applies for a meal.
3. **Textual messages**, used for any free-text message.

The documentation describing the models that need to be supported by each bot endpoints is available here:

http://swagger.u-hopper.com/?url=https://bitbucket.org/wenet/wenet-components-documentation/raw/master/sources/wenet-messages_for_apps-openapi.yaml .

3.2 USAGE

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

The bot is publicly available at https://t.me/wenet_ask_for_help_bot, the use the app the only requirements are: a Telegram account and a WeNet one (that can be created here).

3.3 SOURCE CODE

The source code for both the Eat Together and the Ask for Help bots is tracked on BitBucket and is available here: https://bitbucket.org/wenet/wenet-bots/src/master/. The Readme file includes the description and the step-by-step instructions for correctly running the bots. Also, the steps necessary for building a Docker image of the component are described. The code of the app is released under the Apache 2.0 license.

4. CONCLUSIONS & NEXT STEPS

This deliverable describes the v2.0 of the WeNet platform and the second WeNet app (Ask4Help), which were both used for running the first round of project pilots during Spring 2021.

While the integration in the production environment of the machine learning components (personal and social context builders in particular) is not fully completed yet, the platform in its current version is sufficient for experimenters to build and deploy a fully working application. The material presented in this deliverable is also meant to be used as technical documentation for the WeNet Open Call, which will open in a July 2021. In this context, it will be used to guide applicants and present them a consistent view of the platform capabilities and how the platform can be used to develop and deploy applications.

In terms of enhancements, the following key points represent the milestones for the months to come:

- Complete integration and deployment in the production version of the machine learning components (expected date: October 2021);
- Enhance the structure of the WeNet profile in order to better account for relevant diversity dimensions (joint work with WP1, expected date: October 2021);
- Enable component logging to support troubleshooting and root cause analysis (expected date: November 2021);
- Enhance and extend the Ask4Help chat application for the next round of project pilots. This includes the implementation of additional features and functionality, that will be selected from the outcomes of the exit surveys carried out by WP7 (expected date: November 2021);
- Allow developers to define their own logic in terms of task types, task structure and incentives (expected date: December 2021).

Requests for changes are also likely to occur, as we open the platform for experimentation with non-WP6 applications, either developed by Consortium partners or by the winners of the WeNet Open Call.

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

REFERENCES

[1] D. Miorandi et al., “WeNet platform architecture specifications”, WeNet project deliverable D6.1, 2019.

[2] A. de Götzen et al., “Critical issues and scenarios development”, WeNet project deliverable D7.1, 2019.

[3] D. Miorandi et. al., “WeNet platform and research infrastructure 1.0“, WeNet project deliverable D6.2, 2020.

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