Scalable Understanding of Multilingual MediA (SUMMA)

http://www.summa-project.eu

H2020 Research and Innovation Action
Number: 688139
D6.2 – Release of initial platform

Nature	Report	**Work Package**	WP6
Due Date	30/04/2017	**Submission Date**	30/04/2017
Main authors	Guntis Barzdins (LETA)		
Co-authors	Renars Liepins (LETA), Didzis Gosko (LETA)		
Reviewers	Andreas Giefer (DW)		
Keywords	UX, Microservices, Docker, Cloud		
Version Control			
v0.1	**Status**	Draft	11/04/2017
v0.2	**Status**	Draft	20/04/2017
v1.0	**Status**	Final	26/04/2017

Contents

List of Figures

1 Introduction

The release of the initial SUMMA Platform, Version 1.0, culminates the year-long efforts of all project partners towards the Minimum Viable Product (MVP) to be tested and evaluated by the user-partners of the project: BBC and Deutsche Welle. The initial release of MVP already in the first half of the project is crucial for gathering the feedback from the user partners to guide the targeted development process in the second half of the project.

The architecture of SUMMA Platform is described in the earlier deliverable ”D6.1 - Platform architecture and API Modelling tools selection”; there have not been any deviations from this architecture in the current version of the Platform. New to this version is merely the significant elaboration of two components:

- Now, the user eXperience interface (UX) is based on the Wireframes designed by the user-partners BBC and Deutsche Welle to meet their specific use-case scenarios.
- Now, the most demanding ASR docker module can be distributed over multiple computers (cloud) to handle BigData scaling for near-real-time processing of multiple live video streams.

In this initial SUMMA Platform release, bugs and incompatibilities for most NLP modules have been ironed out to make the whole system sufficiently robust (Liepins et al. (2017a)) in order to move from technology debugging to testing by the end users.

2 User eXperience interface (UX)

The user partners BBC and Deutsche Welle see the SUMMA Platform differently from the technology partners (UEDIN, UCL, IDIAP, Priberam, QCRI, USHEF, LETA) who develop the underlying NLP modules:

- Technology partners focus on their own individual NLP components (ASR, MT, Clustering, NEL, Summarisation etc.) and need detailed performance and debugging information about each individual component. This debugging functionality was provided by the earlier proof-of-concept SUMMA Platform release, described in deliverable "D6.1 - Platform Architecture and API Modelling tools selection"
- User partners assume all NLP technology working flawlessly and instead focus on the additional functionality to organise their actual work-flow, the user-management, the content management, and the intuitive user experience (UX) interface for reduced personnel training requirement. These requirements stem from the use-cases described in deliverable "D1.1 - Use Case Description and Requirements" and were captured in the UX Wireframes (see Appendix) developed by BBC and Deutsche Welle in the Balsamiq[1] UX mockup tool.

2.1 User Interaction with the SUMMA Platform UX

After the initial username/password check, the SUMMA Platform UX opens with the trending screen (Figure 1) as the home screen. This was requested by the user partners to reduce the learning curve for the end-users - rather than asking users to select what they want to do, the available queries with their trending entities are displayed instead. This screen provides a concise high-level overview of "what has been going on" in the news world in the past 24 hours and it also informs the user that Queries are the top level facility for incoming news filtering.

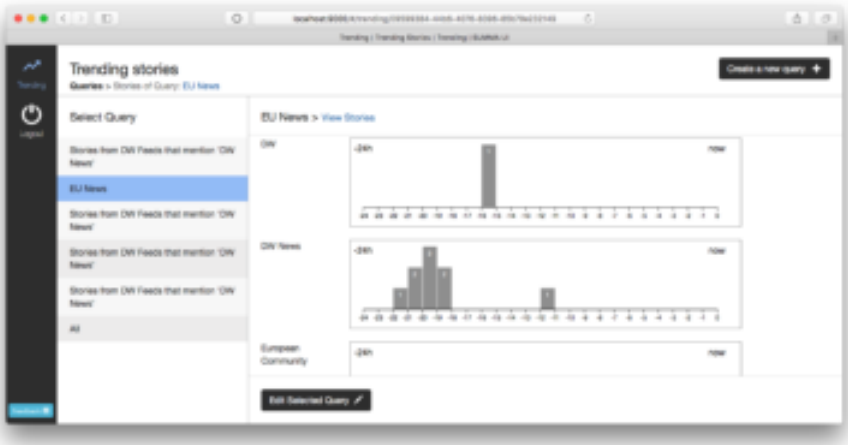

Figure 1: Trending is the home screen of the SUMMA Platform

All SUMMA Platform screens also have a user Feedback button (left bottom corner, distinct turquoise colour) for direct reporting of any bugs or possible improvements in the system.

[1] https://balsamiq.com

Figure 2: Query editing window

Queries can be added and edited as shown in Figure 2. Here the user learns that a Query effectively defines a set of incoming news channels (organised by Feed Groups) to be monitored by the specific query. Optionally, a list of Entities can be added to the Query to narrow it down to monitoring only Stories mentioning any of these Entities of interest. The frequency of mentions for these Entities of interest is what is graphed in the Trending screen to give a high-level news-activity overview.

Figure 3: List of Stories belonging to the selected Query

Users can further drill-down on each query by double-clicking on it. This leads to the Stories screen (Figure 3) showing all Stories belonging to the selected Query. Stories are clusters of MediaItems covering the same or similar news events. The number of MediaItems in the Story and the LastApperance of a new MediaItem in the given Story provide clues to how ”hot” each Story is.

By double-clicking a specific Story its summary, along with a list of MediaItems, is being displayed (Figure 4) from where each individual MediaItem can be viewed (Figure 5) along with

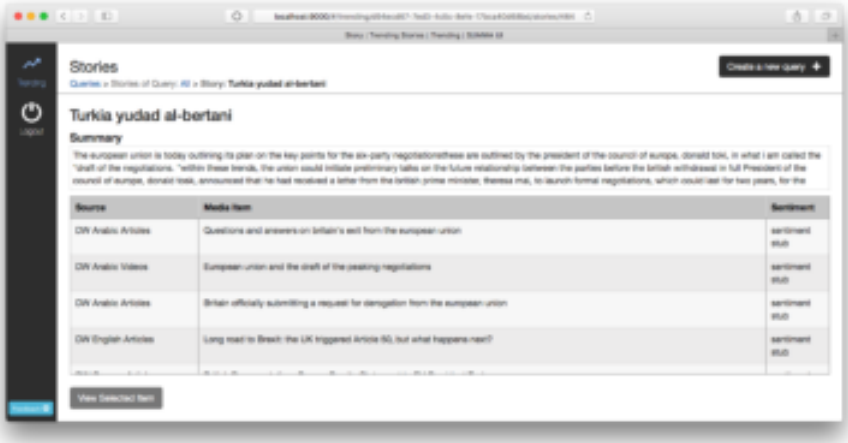

Figure 4: List of Media Items belonging to the selected Story

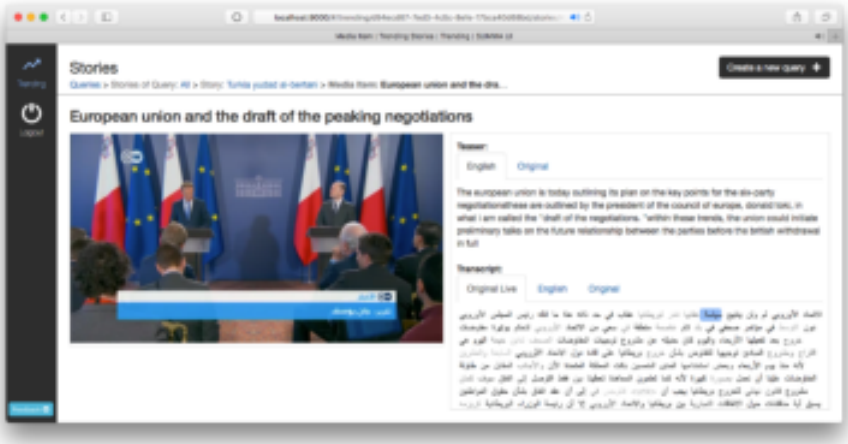

Figure 5: Individual MediaItem view

its transcript, English translation, summary, Named Entities, topics etc. as provided by the NLP components of the SUMMA Platform.

2.2 Administrator Interaction with the SUMMA Platform UX

The users with Administrator privileges, besides having access to the same features as regular users, do have the capability to add/edit/delete news Feeds and FeedGroups (Figure 6, Figure 7), as well as to add/suspend users (Figure 8) on the SUMMA Platform. These operations require Administrator privileges, because they affect all other users of the SUMMA Platform.

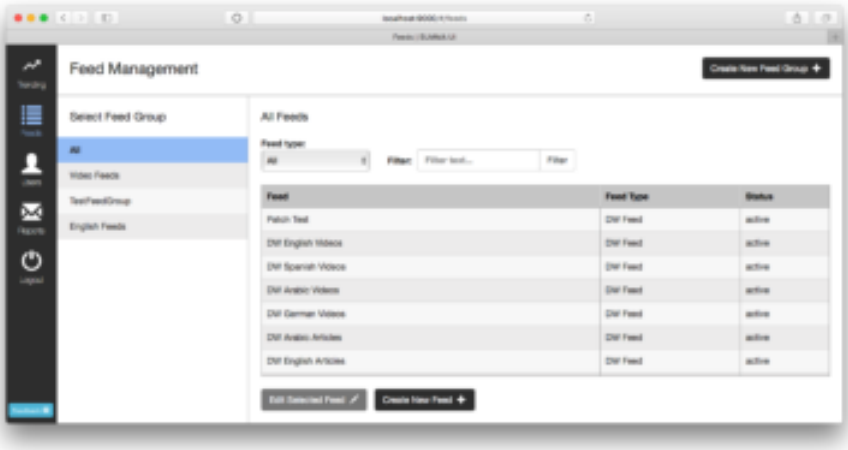

Figure 6: FeedGroups screen. Special FeedGroup "All" shows all ingested Feeds and allows adding/removing feeds (Figure 7) from the ingestor

Figure 7: Feed Edit window allows to add individual feeds and assign them to Feed Groups

Users with administrator privileges also have access to the Feedback Report (Figure 9) , where each Feedback item includes a short description, relevant URL within the SUMMA Platform, and a screenshot of the system at the moment user pressed the turquoise Feedback button to report an issue. Integration of such internal Feedback system within the SUMMA Platform was seen as vital by the user partners DW and BBC, because end users are less likely to provide detailed

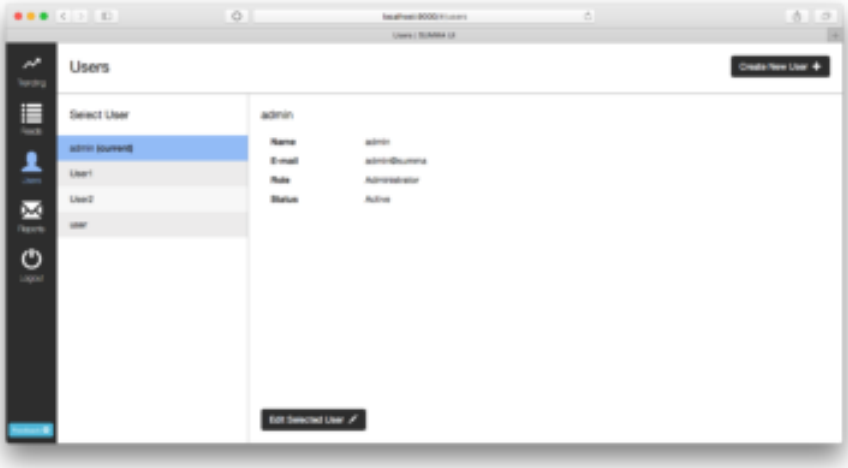

Figure 8: User administration window

Feedback via external channels (e-mail, JIRA) than to press a Feedback button within the system. Additional benefit of this arrangement is that the Feedback button integrated within the system itself has access to ample system state information (screenshot, URL, user) which can be added to the feedback report automatically.

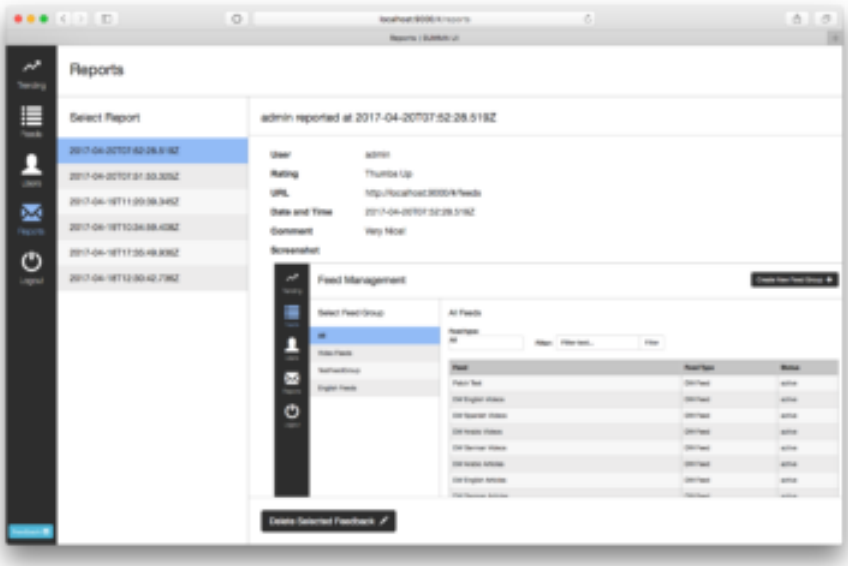

Figure 9: Report on Feedback provided by the SUMMA Platform end-users

2.3 Implementation of the SUMMA Platform UX

UX development was hindered by the fast pace of web programming technology development and associated growth of the end-user expectations for highly dynamic and interactive content presentation while maintaining the backwards compatibility with various web browsers and underlying operating systems.

These high expectations towards UX resulted in the rather steep learning curve towards adoption of the latest web UX development tool Aurelia[2] with built-in Javascript (ES.Next) transpiler to support also older web browsers. The switching from the React.js (used in the earlier proof-of-concept debugging GUI of the SUMMA Platform) towards Angular inspired Aurelia UX technology was warranted also by its support for the two-way data binding and component isolation contributing to the overall UX robustness. The new UX visual design uses the Twitter Bootstrap[3] framework instead of React-specific Material-UI[4] used in the earlier version.

The UX is entirely based on the RESTful API as the "glue" (Figure 10) between the Docker image implementing the actual UX webserver (acting as a proxy for the web client) and the rest of the SUMMA Platform and particularly the Rethink DB at the heart of it. This design not only supports immense flexibility for custom UX integration (like custom dashboards with widgets), but also ensures ultimate integrity for the central Rethink database and independence from the actual database implementation, which might eventually change when the Platform will be scaled up.

The RESTful API "glue" layer is documented using the Postman[5] tool and the actual specification is available also as an internal SUMMA project document Liepins et al. (2017b). For RESTful API documentation, the Postman tool has turned out to be substantially more convenient than Swagger[6] tool used in earlier stages of the SUMMA project. The specification document for "glue" RESTful API shows resource links used to access all SUMMA Platform RethinkDB tables from the UX module along with the actual HTTP request examples. The following HTTP return codes are used:

- 200 – OK, the request (for GETing an entity or entity list corresponding to the requested resource) has succeeded, the resource is sent in the response;
- 201 – Created, the request has been fulfilled and resulted in a new resource being created, the resource is sent in the response;
- 404 – Not Found, the requested resource is not found;
- 409 – Conflict, the request could not be completed due to a conflict with the current state of the DB, e.g. attempting to create user with existing email;
- 422 – Unprocessable Entity, some required fields are missing (for POST and PATCH requests);

[2] http://aurelia.io
[3] getbootstrap.com
[4] material-ui.com
[5] https://www.getpostman.com
[6] http://swagger.io

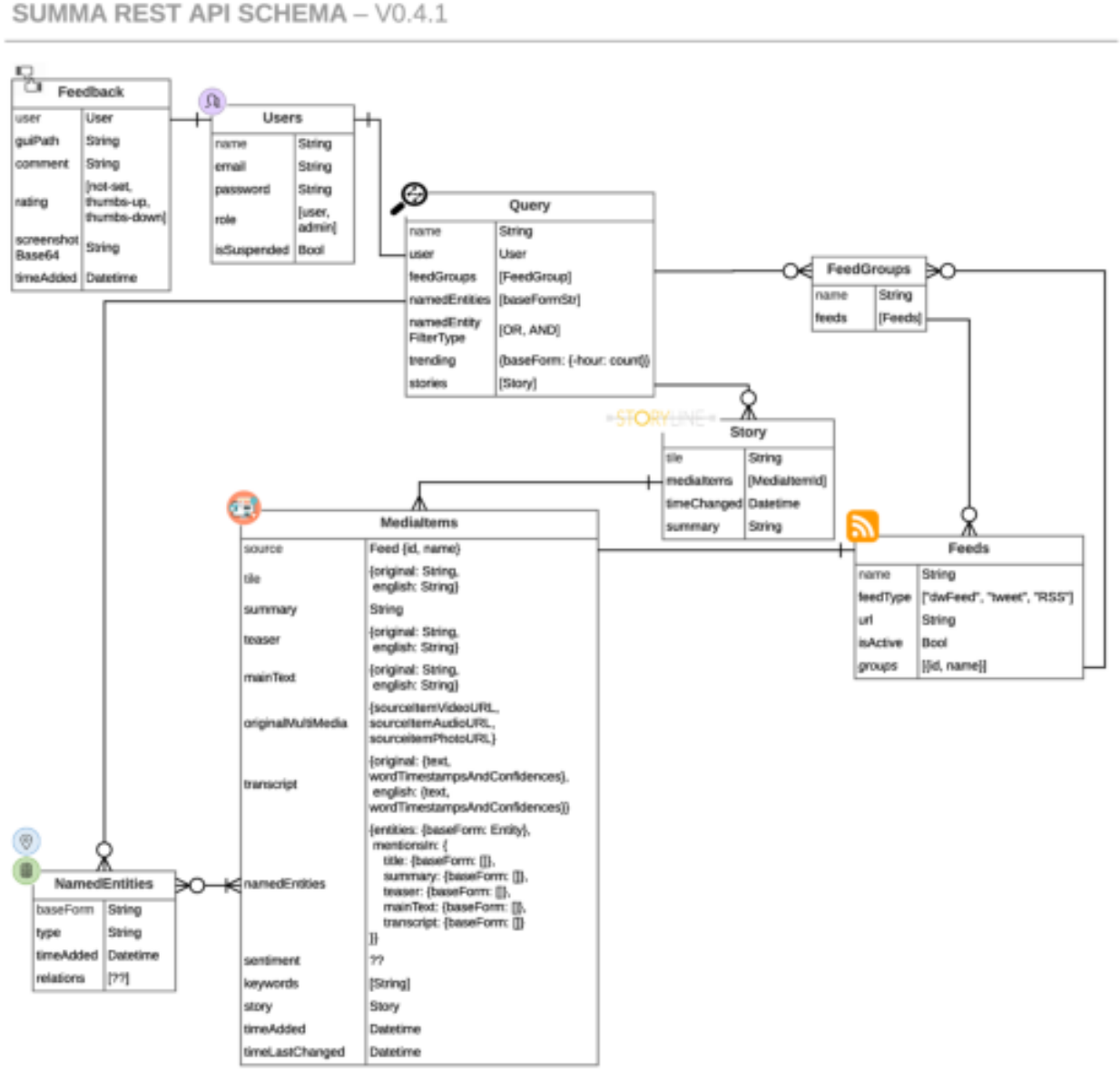

Figure 10: The UX "glue" RESTful API diagram

3 Cloud based BigData scaling

Although true BigData scalability will be addressed in further SUMMA Platform releases, a limited form of Cloud based scalability has been introduced already in the current Version 1.0 of the Platform. This was necessary to accommodate the unexpectedly slow (albeit high quality) performance of the developed Kaldi ASR modules which typically run at only 1/4 speed of real-time, meaning that 4 computers are necessary to process a single real-time audio stream.

3.1 Scaling with Rancher framework

The cloud scalability is enabled by the Rabbit Message Queue (RabbitMQ[7]) - part of the original SUMMA Plattform design. Unfortunately, attempts to make it actually work in the cloud environment ware initially less successful than expected. Although there are numerous cloud-based Docker deployment frameworks such as Rancher[8] (others being Cattle, Swarm, Mesos, Kubernetes), they all come with their own restrictions, requirements, complexity and bugs. Rancher seemed to be the most adequate framework for the SUMMA Platform needs and we spent months of work and considerable Amazon Web Services (AWS) fees (a requirement for Rancher) to get the SUMMA Platform working in this environment. But the resulting system was too complex, restrictive and fragile (e.g. towards naming or components) that we abandoned it in favour of a simpler and more robust solution without dependency on AWS.

3.2 Scaling with Ansible provisioning script

The simpler solution is merely an Ansible[9] provisioning script that uses docker-compose to launch ASR and corresponding wrapper Docker modules on the computers available in the cloud and pointing them to the central publicly exposed RabbitMQ endpoint for job distribution. In the SUMMA Platform context, this nearly trivial solution is as scalable as the mentioned frameworks and has been tested to work flawlessly.

[7] https://www.rabbitmq.com
[8] http://rancher.com
[9] http://ansible.com

4 Conclusion

SUMMA Platform Version 1.0 implements the MVP functionality as specified by the user-partner developed UX Wireframes and is ready for initial user testing and evaluation.

We also conclude that the SUMMA Platform can be implemented in the cloud-based Docker deployment framework such as Rancher, but its use is warranted only for highly heterogeneous architectures and when being deployed in the 3rd party clouds like AWS. In the SUMMA Platform Version 1.0, to speed up ASR, we reverted to a simpler cloud support solution through scripting and a publicly exposed RabbitMQ endpoint.

References

Renars Liepins, Ulrich Germann, Guntis Barzdins, Alexandra Birch, Steve Renals, Susanne Weber, Peggy van der Kreeft, Herve Bourlard, João Prieto, Ondrej Klejch, Peter Bell, Alexandros Lazaridis, Alfonso Mendes, Sebastian Riedel, Mariana S. C. Almeida, Pedro Balage, Shay B. Cohen, Tomasz Dwojak, Philip N. Garner, Andreas Giefer, Marcin Junczys-Dowmunt, Hina Imran, David Nogueira, Ahmed Ali, Sebastião Miranda, Andrei Popescu-Belis, Lesly Miculicich Werlen, Nikos Papasarantopoulos, Abiola Obamuyide, Clive Jones, Fahim Dalvi, Andreas Vlachos, Yang Wang, Sibo Tong, Rico Sennrich, Nikolaos Pappas, Shashi Narayan, Marco Damonte, Nadir Durrani, Sameer Khurana, Ahmed Abdelali, Hassan Sajjad, Stephan Vogel, David Sheppey, Chris Hernon, and Jeff Mitchell. The summa platform prototype. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–119, Valencia, Spain, April 2017a. Association for Computational Linguistics. URL http://aclweb.org/anthology/E17-3029.

Renars Liepins, Didzis Gosko, and Guntis Barzdins. Summa platform ux restful api specification. In *Internal SUMMA Document*, 2017b. URL http://www.ltn.lv/~guntis/summaUXrestAPI.pdf.

5 Appendix - UX Wireframes

Figure 11: User UX Wireframes

Figure 12: Administrator UX Wireframes

ENDPAGE

SUMMA

H2020-ICT-2015 688139

D6.2 Release of initial platform