REVEAL
FP7-610928

REVEALing hidden concepts in Social Media

Deliverable D6.3.2
News Application

Editor(s): Eva Jaho

Responsible Partner: ATC

Status-Version: Final – v1.0

Date: 02/09/2016

EC Distribution: RESTRICTED

Project Number: FP7-610928

Project Title: REVEAL
<table>
<thead>
<tr>
<th><strong>Title of Deliverable:</strong></th>
<th>NEWS application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date of Delivery to the EC:</strong></td>
<td>02/09/2016</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Workpackage responsible for the Deliverable:</strong></th>
<th>WP6 – Framework Integration and REVEAL Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Editor(s):</strong></td>
<td>Efstratios Tzoannos, Stamatis Rapanakis, Eva Jaho</td>
</tr>
<tr>
<td><strong>Contributor(s):</strong></td>
<td>DW, SINTEF</td>
</tr>
<tr>
<td><strong>Reviewer(s):</strong></td>
<td>Linda Rath-Wiggins, Jochen Spangenberg</td>
</tr>
<tr>
<td><strong>Approved by:</strong></td>
<td>All Partners</td>
</tr>
</tbody>
</table>

**Abstract:**
This document is a report accompanying the prototype deliverable of the final REVEAL News Application. It is an updated version of deliverable D6.3.1, after taking into account the evaluation report and updated requirements of deliverable D1.1. In this document, we describe the application targeting the news use-case scenario. Minor modifications will continue on the application until the end of the project including optimization of some system features.

**Keyword List:**
News application, verification tools, media
## DOCUMENT DESCRIPTION

### Document Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Modifications Introduced</th>
<th>Modification Reason</th>
<th>Modified by</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>04/07/2016</td>
<td>Table of contents and draft content</td>
<td>ATC</td>
<td></td>
</tr>
<tr>
<td>V0.2</td>
<td>29/07/2016</td>
<td>Including content from technical partners</td>
<td>ATC</td>
<td></td>
</tr>
<tr>
<td>V0.3</td>
<td>05/08/2016</td>
<td>Update sections and add screenshots</td>
<td>ATC</td>
<td></td>
</tr>
<tr>
<td>V0.9</td>
<td>31/08/2016</td>
<td>Final version after internal review</td>
<td>ATC, DW</td>
<td></td>
</tr>
<tr>
<td>V1.0</td>
<td>02/09/2016</td>
<td>QA – Submission to the EC</td>
<td>INTRA</td>
<td></td>
</tr>
</tbody>
</table>
CONTENTS

EXECUTIVE SUMMARY ........................................................................................................ 7

1 INTRODUCTION .................................................................................................................. 8

2 APPLICATION OF THE FINAL PROTOTYPE ....................................................................... 9
   2.1 ENTRY POINT .................................................................................................................. 9
   2.2 SOCIAL MEDIA STREAM MONITOR ............................................................................. 9
   2.3 MULTIMEDIA BROWSING VIEW .................................................................................. 11
   2.4 SOCIAL MEDIA STREAM ANALYSIS ........................................................................... 13
   2.5 COLLECTION BOARD ................................................................................................ 19
   2.6 VERIFICATION BOARD .............................................................................................. 20

3 SYSTEM CONFIGURATION ................................................................................................. 22

4 CONCLUSIONS .................................................................................................................... 31

5 ANNEX I: USER INTERFACE DESIGN ............................................................................ 32

6 ANNEX II: LIST OF UPDATED REQUIREMENTS ADDRESSED BY RELEVANT MODULES AFTER THE FIRST EVALUATION PHASE ............................................................................. 34
LIST OF FIGURES

FIGURE 1: STREAM MONITOR VIEW ................................................................. 9
FIGURE 2: CREATING A NEW STREAM .......................................................... 10
FIGURE 3: MULTIMEDIA ITEM BROWSER ...................................................... 11
FIGURE 4: INTERACTIVE/DYNAMIC VIEW OF NAMED ENTITIES RECOGNIZED .... 12
FIGURE 5: IMAGE VERIFICATION RESULTS ............................................... 13
FIGURE 6: SENTIMENT DETECTOR ................................................................. 14
FIGURE 7: INTERACTIVE MAP .................................................................. 14
FIGURE 8: INTERACTIVE TIMELINE ............................................................... 15
FIGURE 9: ACTIVE USERS OF A COMMUNITY .............................................. 15
FIGURE 10: INFLUENCE ............................................................................. 16
FIGURE 11: POPULARITY PREDICTION ....................................................... 16
FIGURE 12: COMMUNITY EVOLUTION PREDICTION .................................. 17
FIGURE 13: TOP USERS ............................................................................. 17
FIGURE 14: SEMANTIC SEARCH .................................................................. 18
FIGURE 15: COLLECTION BOARD VIEW ...................................................... 19
FIGURE 16: VERIFICATION BOARD VIEW .................................................... 20
FIGURE 17: TRUTHNest VIEW RESULTS ...................................................... 21
FIGURE 18: FAKE POST DETECTION RESULTS ........................................... 21
FIGURE 19: STREAM MONITOR VIEW .......................................................... 32
FIGURE 20: STORIES (TERMED AS “COLLECTIONS” IN THE APPLICATION) VIEW 33
FIGURE 21: VERIFICATION OVERVIEW ....................................................... 33
# DEFINITIONS, ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
</tbody>
</table>
Executive Summary

Deliverable D6.3.2 represents the final REVEAL News Application (deliverable of ‘Prototype’ nature), which is the application designed and implemented to serve the needs of journalists, or other individuals interested in verifying information that is found in Social Media.

This application utilises the final REVEAL integrated platform that is deployed in a distributed fashion across a large number of servers in the premises of the integrator (ATC), as described in D6.2.2.

We accompany its delivery with this report, which acts as a reference describing the design and implementation of the application explaining the workflows and user interface. It also describes the procedures followed for completing the deployment of the final REVEAL news application.

The news application can be accessed at the following URL: http://reveals.atc.gr/reveal-ui/bs/index.html#/streams.

We would like to point out again that this report is only supportive in nature. The core of the Deliverable is the application “as it stands” (and is continually improved and advanced until the end of the project’s runtime), accessible at the URL quoted above. With this document, we want to provide further information about individual features of the application and how they were derived.
1 Introduction

The approach followed to design and implement the final version of the application, was aligned to rapid prototyping principles. Following the identification of updated user requirements gathered by DW, together with the first round of the evaluation phase, it was ensured that the foreseen application functionalities are both desirable and achievable. These functionalities were designed into a set of wireframes which evolved into a graphical user interface (work done by DW) encompassing workflows that are seen as useful by professionals in the journalism sector, but at the same time achievable by the project’s technical partners. The result of this process is presented in D6.3.1.

In the meantime, the design, implementation and delivery of final individual modules was in progress and monitored by Task 6.2, which ensured their compliance and integration into the back-end platform (D6.2.2) according to the system architecture described in D6.1.1/2. As described in these deliverables, the project is following a continuous integration (CI) approach, which means that every new version of a module is added and technically tested on the platform and can then be used by the application by modifying the front-end and system orchestrator.

In Section 2 we present the final application prototype as it has been achieved and delivered at M34, which however was constantly updated with new and improved functionalities as existing modules were improved, new modules were delivered and evaluation results were received. Minor changes, adjustments and additions will be needed and applied until the end of the project. This is the final iteration in the development cycle of the news application.

The remaining sections of this document are structured as follows:

- Section 2 provides an overview of the usage scenario by the target group of news professionals. It then presents all the functionalities, such as social media stream monitor, analysis, etc.
- Section 3 provides the technical requirements for setting up the application.
- Section 4 concludes this deliverable.

The document also contains two annexes:

Annex I presents the news application graphic interface design provided by DW and reports what have been added in the implementation of the final news application.

Annex II presents the updated requirements (after the first validation phase of the project) that are covered by the existing implementation.
2 Application of the final prototype

In this section we present the final application in the state it is functioning before the contractual delivery of the deliverable (31 August 2016). However, it is imperative to say that improvements are applied on an almost daily basis and new functionalities are released constantly. Therefore, the current status of the application is only up to date in the live version accessible online.

2.1 Entry point

Currently, the entry point of the application is available at http://reveals.atc.gr/reveal-ui/bs/index.html. Authentication is needed to avoid unauthorised access. One user login has been created. The credentials available at this time are:

User name: atc
Password: reveal123

2.2 Social Media stream monitor

When journalists log in, they are able to monitor Social Media streams they have defined in separate columns, and define new ones. They can also access Collections through the thumbnails in a horizontal bar, which are also used to drag and drop stream content into the collections. New collections are defined by selecting a name and image. They are placeholders for manual curation and gathering of content. Posts from a stream can be dragged and dropped into a collection.
New streams are defined by selecting a set of filters:

- a source (currently Twitter and Facebook are supported as they have been defined as the most important sources for journalists in Social Media),
- one or more keyword(s),
- optionally (for Twitter) Twitter username(s) or lists as sources for the stream and all filters provided by the Twitter streaming API (these options change depending on the source).

When a new stream is defined, a new column is created and populated immediately (if Twitter is the source) with content coming from the Twitter search API. A new crawl (assessment) is initiated using the Twitter streaming API which will start to gather content that will be displayed each time the column content is refreshed (by scrolling upwards).

At the top right of the stream column, users can:

- delete the stream

Clicking here deletes the stream
stop the stream from creating other content

get redirected to the multimedia view related to the stream by clicking on the image icon
get redirected to the stream analysis results by clicking on the magnifier icon
filter content in that stream by clicking on the filter icon

2.3 Multimedia browsing view

At this point, a multimedia browsing view has been implemented which allows the user to browse through the collected multimedia associated to that stream, view the entities that have been recognized and detect any manipulation and misuse of web multimedia content.

These functionalities are illustrated in the following figures.

Figure 3: Multimedia item browser
Figure 4: Interactive/Dynamic view of named entities recognized

Entities (from stream “Hillary Clinton”) categorized by type (location, person, organization, other)
Figure 5: Image verification results

The image verification results aims to assist media professionals in the process of detecting multimedia manipulations. Examples of manipulation include maliciously tampering with images and videos, e.g., splicing, removal/addition of elements, while other kinds of misuse include the reposting of previously captured multimedia content in a different context (e.g., a new event) claiming that it was captured there. Furthermore, the multimedia manipulation module integrated in the news application, incorporates a web-based user-friendly interface to make results available to media professionals. For more details on the state-of-the-art algorithms used for detecting multimedia manipulation, please have a look at D3.3.

2.4 Social Media stream analysis

The user can browse more information regarding the stream he/she has selected, as shown in the figures below:

- **Sentiment detector**: analyse the sentiment of the posts contained in the stream
- **Interactive map**: provide an interactive map, relating posts with locations
- **Interactive timeline**: provide an interactive timeline, relating posts with time (horizontal axis) and topics (vertical axis) and level of mentions (indicated by colour)
- **Active communities**: show the communities related to that stream and display more information by clicking on a user of the community
- **Influence**: display information about who posted most about topics related to the stream, filtering with the influence level of the users.
- **Popularity prediction**: provide some information related to the stream selected, such as the number of users that have posted in the stream, number of comments, etc.
- **Community evolution prediction**: predict if the community is evolving or dissolving.
- **Top users**: see the list of top users and information related to them, based on influence level, number of posts and number of followers.

- **Semantic search**: offer to the user an advanced search option, which will return the more semantically similar tweets, rather than the ones that contain only the same keywords, as in the case of a simple search function.
Figure 8: interactive timeline

An exploding thumbnail view from posts that contain a mention of the selected sample time

Figure 9: active users of a community

The users belonging in the community selected (here community no.4) that is being active in the selected stream

Clicking on a user, you get more information, such as connection percentage, no. of tweets, followers and following
Each node represents a user of the selected stream for analysis, and has a radius proportional to its influence, and color according to his main topic of interest.

You can search for influential users, the interactions between them or filter them with regard to the topic of interest.

Figure 10: Influence

Clicking here, a user can select from the list any one of the items that were included in the stream and processed by the module.

The predictions of the popularity values in the four boxes, along with some descriptions of these values.

Click on the left and right arrows to shift the set of features on display.

Using the bar below the popularity values, the user may select the particular lifetime of the posts.

Figure 11: popularity prediction
Figure 12: Community evolution prediction

- Number of users in the community
- Community's topics
- Users' ids
- Community evolution prediction, in this case "dissolving"

Figure 13: Top users

- Get top users by influence, posts or followers
- Clicking on a user you get his profile on the right
- The profile of the user selected showing his topics of interest in the stream
- The posts of the selected user in the stream
The user provides the system with a phrase that s/he wants to find more information.

After entering the phrase, you get a list of posts that contains all related relations.

Filter arguments based on entities involved (person, organization or location).

Filter subjects based on entities involved (person, organization or location).

Figure 14: semantic search
2.5 Collection Board

Each collection is a container of content manually added from the stream monitor view. Users can create new collections or delete existing ones. They can sort it by various criteria and also select an item and, by clicking on the 'verification status', go to the Verification Board.

![Collection Board View](image)

Figure 15: Collection board view
2.6 Verification Board

The Verification Board is called when users want to verify a specific Social Media item. This board includes three sections:

- **The Content**: containing basic information about the post
- **Verification checklist**: a section that contains manually editable annotation fields. Automatically computed information can be manually added to this section from the third section ("What Reveal found out" as shown in the figure below).
- **What Reveal found for you**:
  - **Truthnest**: Information received by the API of the ATC TruthNest application¹, which provides verification information at individual tweet level.
  - **Fake Post Detector**: Information regarding the veracity of the tweets that have been computed in relation to content and contributor features of historic cases of fake and real tweets.
  - **Similar Posts**: a view of similar posts related to that item.

---

¹ Information about the TruthNest application is available under [http://www.truthnest.com/](http://www.truthnest.com/)
By clicking on each category above (Contributor, Content, Context) the user can get more details about an item and its creator.

By clicking on each of the parameters on the left (based on features and metadata extracted from the post and its contributor), the selected post is classified as fake or real depending on whether it contains misleading multimedia content.

Figure 17: TruthNest view results

Figure 18: Fake post detection results
3 System configuration

In this section we describe the guidelines to setup the environment and the REVEAL modules for running successfully the news application.

Install Java in any Ubuntu Server that is hosting Java applications:

$ sudo add-apt-repository ppa:webupd8team/java

$ sudo apt-get update

$ sudo apt-get install oracle-java8-installer

Install mongoddb:

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

Install Docker:

https://docs.docker.com/engine/installation/linux/ubuntulinux/

Install Apache2 Web Server:

https://httpd.apache.org/docs/current/install.html

Or by using Ubuntu apache2 repo:


Install Apache Tomcat 7:


Install RabbitMQ:

Note: we could skip this step if we use the RabbitMQ instance included in ITINNOs Social Media Client Framework Docker image

https://www.rabbitmq.com/install-debian.html

Deploy Main User Interface (AngularJS) in Apache2 web server:

Gitlab account for code:

- **Username**: reveal_official
- **Password**: revealatc
1. Download project from: [https://gitlab.atc.gr/reveal_internal/reveal-ui.git](https://gitlab.atc.gr/reveal_internal/reveal-ui.git)
2. Deploy the downloaded project in default apache2 directory (/var/www/html)
3. Configure the location of generic-web-services (host, port)
4. Restart Apache: `$sudo service apache2 restart`

**Deploy orchestrator in Apache Tomcat 7:**

1. Download and build from: [https://gitlab.atc.gr/reveal/orchestrator.git](https://gitlab.atc.gr/reveal/orchestrator.git)
2. Deploy the produced war file found in /target folder of the project to /var/lib/tomcat7/webapps directory in the Ubuntu server where tomcat 7 is running.
3. Restart Apache Tomcat 7: `$sudo service tomcat7 restart`

**Deploy orchestrator client:**

Deploy orchestrator client based on architecture described in section “Offline Process Orchestrator” of D6.2.3:

More precisely, each server containing offline analysis modules must contain an instance of orchestrator client. This means that for each server, we have to:

1. pull the code from: [https://gitlab.atc.gr/reveal/controller.git](https://gitlab.atc.gr/reveal/controller.git)
2. change `/src/main/resources/broker.properties`
   
   ```
   exchange_name=<offline-modules-host>  
   broker_host=<rabbitMQ-host>  
   broker_username=  
   broker_password=
   ```
3. rebuild the project with any IDE.
4. Then, deploy the output of the project (`controller-1.0-SNAPSHOT-jar-with-dependencies.jar`) found in /target directory of the project to the hosting server.
5. execute it with: `$java --jar controller-1.0-SNAPSHOT-jar-with-dependencies.jar`

We need to use the `<offline-modules-host>` name when building the central orchestrator to trigger the execution of the offline modules in each hosting server.

**Deploy mq2mongo:**

- Download code from: [https://gitlab.atc.gr/reveal/mq2mongo.git](https://gitlab.atc.gr/reveal/mq2mongo.git)
- Change properties file inside project and build it with any IDE (eclipse, netbeans, IDEA)
- Get the file `mq2mongo-1.0-SNAPSHOT-jar-with-dependencies.jar` from the /target directory of the project and deploy it on any server running Java
- Start it with `$java -cp ../mq2mongo-1.0-SNAPSHOT-jar-with-dependencies.jar eu.reveal.mq2mongo.Main`
Deploy generic-web-services:

1. Pull the code from the git repo: https://gitlab.atc.gr/reveal/generic-web-services.git
2. Configure the properties file with `<mongo-host>`, `<mongo-port>`
3. Build the project with any IDE
4. Deploy the output found inside `/target` directory of the project inside Apache Tomcat 7 (`/var/lib/tomcat7/webapps`)
5. Restart Apache Tomcat 7

Deploy Multimedia Crawler:

Find code and instructions at github: https://github.com/MKLab-ITI/reveal-media-webservice

Deploy offline analysis modules:

1. Download the archives found inside: http://squall-6.atc.gr/offline-modules/
   - **username**: reveal
   - **password**: r3v3@l
   
   Particularly for the influence (NokiaLabs and NCSR) you can download the code from:
   - https://gitlab.atc.gr/reveal/influence/tree/alblf (NokiaLabs - java module)
   - https://gitlab.atc.gr/reveal/influence/tree/gensim (NCSR - python module)

   To execute the python module as: `sudo python integration.py <assessment_name> items twitter yes influencers`

   To execute the java module: `java -jar UnsupervisedInfluence.jar <assessment_name> items twitter`

2. Follow the steps included in the README file of each zipped archive.

Deploy online analysis modules (Docker images):

First Download all the Docker images found in Docker registry – in order to connect to registry follow the steps below:

**Step 1: Download a pre-built image**

```
# Download an ubuntu image

$ sudo docker pull Ubuntu:12.04
```

This will find the `ubuntu` image by name on Docker Hub and download it from Docker Hub to a local image cache.
**Step 2: Running an interactive shell**

# Run an interactive shell in the ubuntu image,
# allocate a tty, attach stdin and stdout
# To detach the tty without exiting the shell,
# use the escape sequence Ctrl-p + Ctrl-q

$ sudo docker run -i -t -name <container_name> Ubuntu:12.04 /bin/bash

**Step 3: Listing containers**

$ sudo docker ps # Lists only running containers

List running containers to verify it's running.

**Step 4: Attach**

Usage: docker attach [OPTIONS] CONTAINER

Attach to a running container

--no-stdin=false Do not attach STDIN

--sig-proxy=true Proxy all received signals to the process (even in non-TTY mode). SIGCHLD, SIGKILL, and SIGSTOP are not proxied.

Example:

$ sudo docker attach <container_id> OR <container_name>
The `attach` command will allow you to view or interact with any running container, detached (`-d`) or interactive (`-i`). You can attach to the same container at the same time - screen sharing style, or quickly view the progress of your daemonized process.

You can detach from the container again (and leave it running) with `CTRL-p` `CTRL-q` (for a quiet exit), or `CTRL-c` which will send a SIGKILL to the container, or `CTRL-\` to get a stacktrace of the Docker client when it quits. When you detach from the container's process the exit code will be returned to the client.

To stop a container, use `docker stop`.
To kill the container, use `docker kill`.

**Step 5: Committing (saving) a container state**

Save your containers state to an image, so the state can be re-used.

When you commit your container, only the differences will be stored (as a diff). See which images you already have using the `docker images` command.

```
# Commit your container to a new named image
$ sudo docker commit <container_id> <some_name>

# List your containers
$ sudo docker images
```

**Step 6: Account creation and login**

Typically, you'll want to start by creating an account on Docker Hub (if you haven't already) and logging in. You can create your account directly on Docker Hub, or by running:

```
$ sudo docker login
```

This will prompt you for a user name, which will become the public namespace for your public repositories. If your user name is available, Docker will prompt you to enter a password and your e-mail address. It will then automatically log you in. You can now commit and push your own images up to your repos on Docker Hub.

**Step 7: Contributing to Docker Hub**
Anyone can pull public images from the Docker Hub registry, but if you would like to share your own images, then you must register first, as we saw in the first section of the Docker User Guide.

**Pushing a repository to private REVEAL Docker Registry**

In order to access the Reveal Docker private registry (https://reveal.atc.gr:5001) please follow the instructions:

1. Create the certificate directory in your client machine:
   
   ```
   $ mkdir /usr/local/share/ca-certificates/docker-dev-cert
   ```

2. Copy the SSL Certificate in the certificate directory

3. Update the certificates:
   
   ```
   $ update-ca-certificates
   ```

4. Restart Docker to make sure it reloads the system's CA certificates:
   
   ```
   $ sudo service docker restart
   ```

*Make sure that you have saved any changes in your containers before you restart them.*

Now you should be able to log in to REVEAL private registry:

```
$ docker login https://reveal.atc.gr:5001
```

You may use the following credentials:

**Username:** reveal-dev

**Password:** r3v3al

In order to come up with an image out of your Docker container:

1. commit any changes:
   
   ```
   $ docker commit $[container_Id] $[image-name]
   ```

2. You have to tag an image with the private registry's location in order to push to it:
   
   ```
   $ docker tag $[image-name] reveal.atc.gr:5001/$[image-name]
   ```

3. push that image to our registry:
   
   ```
   $ docker push reveal.atc.gr:5001/$[image-name]
   ```
Docker Image Configuration:

**OpenStreetMap Docker image**

The only configuration needed is couple of edits to a configuration file.

1. Start docker image:
   - `docker run -it reveal.atc.gr:5001/reveal-social-media-client-framework_v1_4.atc.gr /bin/bash`
2. Open main social media client framework configuration file:
   - `nano /var/lib/reveal/social-media-client-framework/config/social_media_client_framework.ini`
3. Edit:
   - `client_framework_host`
   - `client_framework_port`
   - `rmq_host`
   - `rmq_port`
   - `psql_username`
   - `psql_password`
   - `psql_host`
   - `psql_port`
   - `psql_db_name`
   - `psql_db_schema`
4. Commit changes back to the image:
5. Run the Docker image:
   - `docker run -it <expose any ports here or bind it to your host> reveal.atc.gr:5001/reveal-social-media-client-framework_v1_4.atc.gr sh /var/lib/reveal/run_no_vnc.sh`

**DSS Docker image:**

1. Start docker image:
   - `docker run -it reveal.atc.gr:5001/reveal-dss-framework_v1_2.atc.gr /bin/bash`
2. Open main DSS backed controller configuration file who will be serving DSS HTML pages, making DB queries etc.
   - `nano /var/lib/reveal/dss-backend-server/config/dss_backend.ini`
3. Edit:
   - `dss_backend_host`
   - `dss_backend_port`
   - `dss_backend_resource_dir_simplehttpserver_scheme`
   - `dss_backend_resource_dir_simplehttpserver_host`
   - `dss_backend_resource_dir_simplehttpserver_port`
   - `db_user`
   - `db_pass`
   - `db_host`
   - `db_port`
4. Commit changes back to the image:
   - `docker commit $[container_id] reveal.atc.gr:5001/reveal-dss-framework_v1_2.atc.gr`
5. Run the Docker image:
   - `docker run -it <expose any ports here or bind it to your host> reveal.atc.gr:5001/reveal-dss-framework_v1_2.atc.gr sh /var/lib/reveal/run_no_vnc.sh`
Storm and Geospatial Controller Docker image

1. Start docker image:
   - docker run -it reveal.atc.gr:5001/zookeeper_nimbus_topologies_controller.atc.gr /bin/bash

2. Open main geospatial controller configuration file:
   - nano /var/lib/reveal/geospatial-controller/config/geospatial_controller.ini

3. Edit:
   - geospatial_controller_host
   - geospatial_controller_port
   - smcf_host
   - smcf_port
   - rmq_host
   - rmq_port
   - rmq_username
   - rmq_password
   - rmq_queue_name
   - db_user_controller
   - db_pass_controller
   - db_host_controller
   - db_port_controller

4. Now open thumbnail creation application configuration file:
   - nano /var/lib/reveal/geospatial-controller/config/thumbnail_creation_app.ini

5. Edit:
   - db_user
   - db_pass
   - db_host
   - db_port

6. Regenerate the related topology jar files by replacing the following values:
   - RabbitMQ hostname
   - RabbitMQ port number
   - RabbitMQ username
   - RabbitMQ password
   - PostgreSQL username
   - PostgreSQL password
   - PostgreSQL host
   - PostgreSQL port
   - RabbitMQ exchange for CERTH
   - RabbitMQ exchange for NOKIA
   - RabbitMQ exchange for UKOB

7. Send me all configurations from the above step and I will happily exchange it for ready to use topology jars

8. Delete all previous topology jars (we want to avoid any ambiguities during their replacement)
   - rm /var/lib/reveal/geospatial-controller/topology-jars/*.jar

9. Copy topology jars that I will send you:
   - docker cp <folder>/*.jar <controller container id>:/var/lib/reveal/geospatial-controller/topology-jars

10. Commit changes back to the image:
    - docker commit $[container_id]
        reveal.atc.gr:5001/zookeeper_nimbus_topologies_controller.atc.gr

11. Run the Docker image:
    - docker run -it <expose any ports here or bind it to your host>
        reveal.atc.gr:5001/zookeeper_nimbus_topologies_controller.atc.gr
Initial Docker Image Testing and Debugging:

Start all the docker images in a sequence as specified above i.e. 1) Database image, 2) Social Media Client Framework image, 3) DSS and 4) Storm with Geospatial Controller

Social Media Client Framework Docker image

4 Conclusions

The first evaluation results enabled the second iteration of requirements, specifications and development. This deliverable is an update of D6.3.1 containing requirements implemented as functionalities. This deliverable aims at describing the final version of the REVEAL news application. It presents the news system features and interfaces. Screenshots of relevant (but not all) features were included in order to depict the implemented functionality (we have presented the status in which the application is one week before the contractual delivery of this deliverable).

In the development of the news application we have had continuous feedback rounds by Deutsche Welle journalists and media professionals on the UI and the functionality of the application. Next to that there has also been an evaluation on the static interface by SINTEF with external users. Some of the findings and insights of these evaluations also found their way into the current interface. The proposed graphic interface (see Annex I) by DW was a good basis to work from and provided room for all modalities of the REVEAL project to be integrated easily.

Finally, minor adjustments and changes will take place according to the feedback from the final evaluation phase. Based on the final / current REVEAL news application, the final evaluation phase will follow, deriving the evaluation results that will be described in D7.2.
5 Annex I: User Interface design

The latest version of the UI design after iterations following formative evaluation sessions organised by DW and SINTEF was presented in D6.3.1. The design has been produced by DW (using Axure RP7 Pro) as a clickable prototype transforming the requirements depicted in the conceptual and statical UI into an application. This clickable prototype also functioned as a blueprint for ATC to implement the functional online prototype. In this last version of the news application, we have followed the same design as depicted in the figures below (Fig. 19-21) for the stream monitor view, the collection board and the verification page (for more details on design screenshots please refer to D6.3.1). What we have added in the implementation of the last version of the news application, is the stream analysis page (section 2.4) in which we bring different results related to the stream selected by the user, whereas in the verification page we display results related to the selected item of the stream (section 2.6). Moreover, we added a multimedia browsing view page if the user would like to browse multimedia related to the selected stream (section 2.3).

![Stream Monitor View](image-url)
Figure 20: Stories (termed as “Collections” in the application) view

Figure 21: Verification overview
6 Annex II: List of updated requirements addressed by relevant modules after the first evaluation phase

In the first version of the news application (D6.3.1), we addressed requirements derived from D1.1 based on the modules that were available by that time. After the first evaluation phase (results reported in D7.1), in this final version of the news application we have addressed the updated requirements (as reported in D1.1a). Here we present this list of requirements (Dx, Ax, and Bx refer to the respective requirement number as referenced in D1.1a), which were not fulfilled by the first version of the application, and now have been successfully addressed by the relevant modules added in the REVEAL platform (Mx refers to the respective module number as referenced in D6.2.2).

- D10: I receive a list of related topics based on my search terms
  - M24 - NCSR - When editing the stream query, top related entities will be suggested
- D11: I can receive a collection of users derived from the networks of users I have defined as trustworthy
  - M6 - ITINNO - Will be visualised by the trust module
- D13: I can follow my trusted network
  - M21 - ITINNO/ATC - Will be added as a feature in the “create stream” functionality
- D14: I can define parameters for relevance/influence that will have impact on search
  - UI - ATC – Will be implemented by the stream view filtering
- A1: I can search for contributors based on search terms – independent on which platform they are on
  - M22 - ATC/NCSR - Each stream is based on search terms. Contributors within streams are sorted as influencers or members of topic-based communities
- A2: I can define and share a trusted network based on a topic
  - M36 - ITINNO - The trust module will allow and visualise this
- A6: I can receive information about the community the contributor is located in
  - M8/M16/M22 CERTH/NCSR - Communities will be visualised in the stream view and associated topics discussed will be shown
- A7: I can see who posted the most retweeted tweet based on search terms
  - UI - ATC - By sorting feature on the stream column
- A8: I can see who posted most about a topic based on search terms
  - M22 - NCSR/ATC - By sorting/filtering features on the stream column using the topic based influence
- A9: I can determine the most influential contributors - the way influence is calculated should be transparent
  - M22/M2 NCSR/NOKIA/ATC - By sorting/filtering features on the stream column
- A10: I can find the most relevant links posted by the most influential contributor
  - M2/M6 NOKIA/ITINNO – Calculated by M2 and visualised by M6
- A11: I can identify the most relevant photos posted by the most influential contributor
  - M2 NOKIA/ATC – Filtering on the stream will show high influencers who posted photos
- A13: I can find the most relevant topics (e.g. other #malaysia #airplane) posted by the most influential contributor
  - Same as A11
- A15: I can filter influential contributors based on # of followers, # of tweets
  - UI - ATC - By sorting/filtering features on the stream column
- A16: I can see what my trusted network is talking about
  - M20 ITINNO Visualised by M20
- A17: I can find various communities discussing different perspectives of one story
  o M22 – NCSR - Communities will be visualised in the stream view and associated topics discussed will be shown
- A18: I can identify the sources who posted the original posts as opposed to just “retweets”
  o M33 - NOKIA - Visualised as a stream in the verification view
- A19: I can create a list of trusted users for a specific story/topic
  o M36 - ITINNO – To be defined and visualised by M36
- A20: I can manually define who is trustworthy/ influential
  o ITINNO - Trusted users will be manually defined (not through the system)
- B10: I can search for most relevant text based on keywords AND/OR based on trusted user
  o M6 - ITINNO – Will be visualised through the WP5 framework
- B12: I can search for similar text
  o M33 – NOKIA - Source finding module will be based on this