Rights, Equality and Citizenship (REC)

Programme of the European Commission

(2014-2020)

Monitoring and Detecting Online Hate Speech

D3.3: Smartphone app[†]

Abstract: The aim of this document is to present the implementation of a hate-speech reporting Smartphone app, as well as to be used as a how-to-use guide. It describes in details the reason behind each design and implementation choices, as well as the whole design process. Each of the application's views is described and the techniques used for better user experience are explained.

Contractual Date of Delivery	March 2017
Actual Date of Delivery	May 2017
Deliverable Security Class	Public
Editors	D. Stefanidis, D. Paschalides, G. Pallis
Contributors	All *MANDOLA* partners
Quality Assurance	

[†] This project is funded by the Rights, Equality and Citizenship (REC) Programme of the European Commission.

The *MANDOLA* consortium consists of:

FORTH	Coordinator	Greece
ACONITE	Principal Contractor	Ireland
ICITA	Principal Contractor	Bulgaria
INTHEMIS	Principal Contractor	France
UAM	Principal Contractor	Spain
UCY	Principal Contractor	Cyprus
UM1	Principal Contractor	France

Document Revisions & Quality Assurance

Internal Reviewers

Revisions

Version	Date	By	Overview
0.1	04 March 2017	D. Paschalides, D. Stefanidis	Smartphone app
0.5	07 March 2017	D. Paschalides, D. Stefanidis, G. Pallis, M. Dikaiakos	Smartphone app
1.0	11 March 2017	D. Paschalides, D. Stefanidis, G. Pallis, M. Dikaiakos	Smartphone app
1.1	16 March 2017	D. Paschalides, D. Stefanidis, G. Pallis, M. Dikaiakos	Smartphone app
1.2	22 March 2017	D. Paschalides, D. Stefanidis, G. Pallis, M. Dikaiakos	Smartphone app

Table of Contents

Table of Figures

1 Introduction

Hate speech, as described and analysed in previous deliverables, is currently overwhelming the online communities and especially the social networks. Companies such as Facebook, Twitter, Google and YouTube are under increasing scrutiny for the amount of hate speech that thrives on their platforms during the recent years [1] [2] [3] [4]. However, they don't have a comprehensive response for dealing with hate speech due to the fact that there is a fine line between free and hate speech, and they don't want to limit freedom of expression on their services. Through time they have issued measures against it, but while the steps were positive, they couldn't eliminate the problem of hate speech.

Facebook, Twitter, Google, YouTube and Microsoft have all been involved in the creation of the "code of conduct" to fight illegal hate speech on their platforms within 24 hours. The EU asked that companies be willing to disable or remove content from their platforms and if necessary, to promote "'counter-narratives' to hate speech," Reuters reported. But the European Commission revealed that the companies were not complying with this code in a satisfactory manner.

Reporting online hate speech is a crucial part in order to address this matter, and there are a lot of governmental and non-governmental organizations providing web portals in order for citizens to report hate speech. But considering that nearly 80 percent of the social media time is spent on mobile devices, using such web portals is difficult on such devices, thus users tend to ignore reporting any hate speech encounters. Statistics show that roughly 1 in 5 minutes of all digital media time is spent on social sites or in social media apps across the desktop and PC [5].

In this deliverable, the design and implementation of the Smartphone app is presented, which has been developed in order to provide citizens with a user friendly mobile application for easier hate speech reporting, by taking into account user's anonymity. The application will provide: **i)** the ability of anonymous reporting of hate-related speech and material found in the web and social media; **ii)** compatibility with Android, IOS and Windows mobile devices covering more than 99% of the market as depicted in Figure 1; **iii)** statistical analysis considering hate speech in order to raise the awareness of the user about the impact of hate speech on the world and **iv)** a FAQs section. The rest of the deliverable is structured as follows: Section 2 presents the architecture of the mobile application and the reporting infrastructure. The design and development phase is presented in Section 3. The Section 4 presents the main features of the API. Section 5 presents the user interface and views. In Section 6 there will be a more detailed analysis on the reporting mechanisms that were implemented, and finally the Section 7 will conclude this deliverable.

2 Reporting Application's Architecture

The Smartphone app for mobile devices consists of the following two main modules:

- The **awareness module**, where static data is presented to the user in order to raise his/her awareness on the online hate speech matter;
- The **anonymous reporting module**, which provides the users with methods in order to report hate speech encounters anonymously.

The Smartphone app provides two methods for reporting hate speech. The first one is via the **MANDOLA Proxy server**, which uses a way back machine [6] and loads the selected URL in a web browser within the application (referred to as InAppBrowser) in order for the user to highlight the hate speech content. The MANDOLA Proxy is mainly used for public encounters of hate speech in sources such as YouTube, Twitter, news sites and forums, where the content can be viewed publicly and the user is not required to provide any personal information. The second method for reporting hate speech is through the **OCR - Optical Character Recognition**. While the user is browsing with his/her private social media accounts via their native applications, he/she can take a screenshot and report the hate speech encounter without providing any personal information.

The above two methods, OCR and MANDOLA Proxy, have been developed in order to preserve the user's anonymity and make the reporting easier without reducing the user experience in the social media native applications. Thus, Smartphone app runs as a **background process** where it listens for a "URL Copy" event and automatically generates the MANDOLA hate speech report, in order to provide a smoother and easier user experience. The submitted reports are also **stored locally** in a **SQLite database**, so as to provide the ability to the user to view and analyze any previous reports. Via the **MANDOLA API** these reports are also stored into the **Report Storage Module**, which is an encrypted and secure relational database for keeping the hate speech reports. Figure 1 depicts the reporting application architectural diagram. The developed API can be utilized by the appropriate authorities to provide them with these reports in order to address them properly.

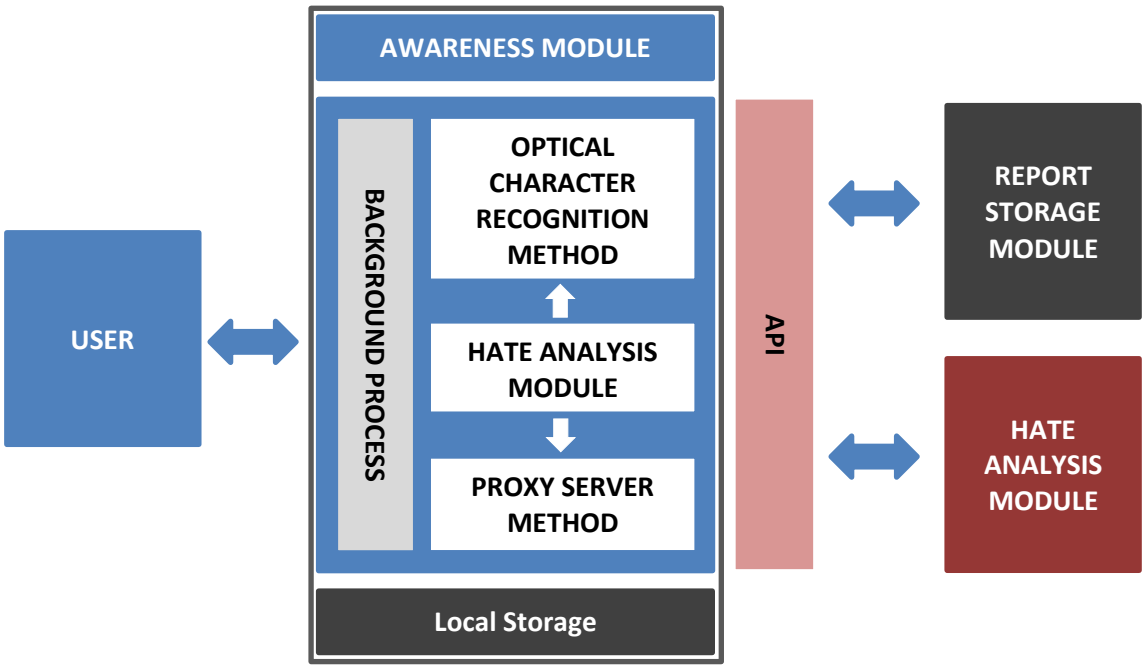

Figure 1: Smartphone app architectural diagram.

3 Design

3.1 Design Concepts

According to [7], a reporting application should 1) protect the user's privacy, 2) be able to provide the user with information regarding the impact of online hate speech, and 3) be easy to use without reducing the user's experience. Based on these three concepts the design of the application was implemented by using mock-ups, and the basic look-and-feel of the smartphone app was established. In the next subsection, the design thinking for each of those concepts is described.

3.1.1 User's Anonymity

One of the key factors of the smartphone app is user anonymity. By providing users with an anonymity privilege, they are secured from the possibility of revenge and serious misuse, avoid persecution for their beliefs, opinions and views on political, religious and other matters. The users of the application do not need to provide credentials to access it. By providing safe reporting of online hate speech incidents, users are encouraged to report and initiate actions against such incidents.

The reports are divided into two types: a) the public reports, meaning reports from public sources accessible from everyone without the need to login, and b) the private reports, meaning reports from within user's private social media accounts. For the public reports, the user must first load the URL from within the smartphone app. Afterwards, the user can highlight and report hate speech. For the private reports, an Optical Character Recognition (OCR) module was integrated, where the user can take a screenshot of the hate speech contained in the text within a social media native application, without requiring him/her to provide any credentials. By loading the screenshot and selecting the area containing hate speech, the OCR method will recognize the text and generate the report. The workflow after the text recognition is the same as with the public reports.

3.1.2 User's Awareness

Another crucial objective of the smartphone app is to raise user awareness on the problem of online hate speech. The Smartphone app tries to fill this gap by providing facts, statistics on hate speech, analytics on events influencing hate speech and also providing the FAQs, based on the D.4.1 of MANDOLA [8].

3.1.3 User Experience

A factor for the smartphone app's success is user experience. Therefore, there is a need to adopt a user-centered approach to mobile application development that gives emphasis to the needs of the users. For the Smartphone app, the user experience and the usability of the reporting system is the key in order to encourage people to report online hate speech. This is why the methodology and steps of the reporting system were designed to be simple. The user is required to fill four main fields in order to report any hate speech encounters: the source

URL, which is mandatory, in order for the moderator to view and validate the report, the hate speech containing text, the categories of the hate speech content, and an optional title.

To further improve the user experience, the **MANDOLA Bubble** has been created. The MANDOLA Bubble is a background process of the Smartphone app that can be viewed as a bubble head (e.g. Facebook Messenger). While user is browsing on the mobile browser or other social media applications and encounters possible hate speech, he/she copies the URL and automatically the MANDOLA Bubble senses this and asks the user if he/she would like to report it and automatically generates the report. This way the user experience becomes easier, smoother and more encouraging to the user.

3.2 Application's Mock-ups

In order to establish a prototype of the application and an image of what is going to look like, mock-ups have been created based on the objectives of the Smartphone app (depicted in Figure 2).

3.2.1 Awareness and FAQs Mock-up

Awareness is the application's view where facts and statistics are presented to the user in order to raise his/her awareness on the impact of online hate speech to the community. It will be a scrollable interface where multiple charts, facts and statistics will be displayed. This view can also be connected with the MANDOLA Monitoring Dashboard [9] in order to provide the users with dynamic statistical analysis on hate speech. There is also the FAQs view, where the questions and answers from D.4.1 are presented in a more compressed form. The user will be able to view each question's answer within an accordion like component, and also search a question based on keywords.

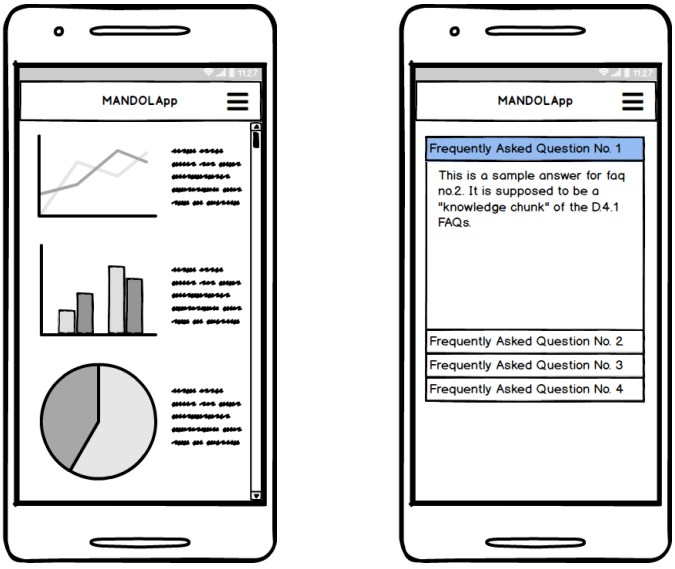

Figure 2: Awareness (left) and FAQs (right) views mock-ups.

3.2.2 Reporting Mock-ups

The reporting in smartphone app has several views and interfaces including the list view, where all the reports of the user are shown, the info view presenting information on a specific report, and the steps for a report to be submitted.

For the list view, as it can be seen in Figure 4, the reports are listed in descending order based on the date they were submitted. Each list item shows a logo to identify the URL host, the report's title (if there is one), a snapshot of the content and the date. There is also the functionality to manually create a new report by pressing the floating button with the plus icon, where the user can select one of the two main reporting methods.

By pressing one of the list items, the information view (depicted in Figure 3) of the specific report is presented. In this view the report is shown in more details containing the title, if provided, the exact date and time of the report submission, the actual text and the categories of the hate content as they were selected by the user. From here it will be possible for the user to load the specific URL into the browser in order to view the content.

The reporting steps, as they were described above, consist of giving the source URL, an optional title of the report, the hate speech containing text and the categories of the hate speech content. These steps, as seen in Figure 5, will be in separate windows with a button "Next" connecting each other. At the end, a button "Submit" will appear to submit the report. These fields will be automatically filled if the user uses the MANDOLA Bubble.

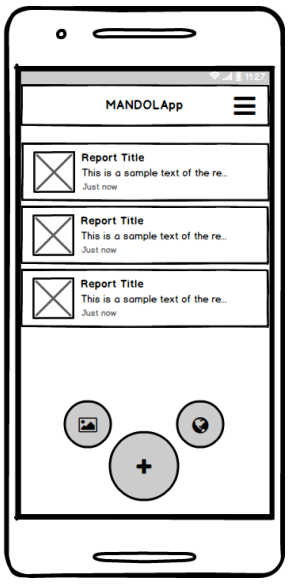

Figure 4: Report view list mock-up.

Figure 3: Report information view mock-up.

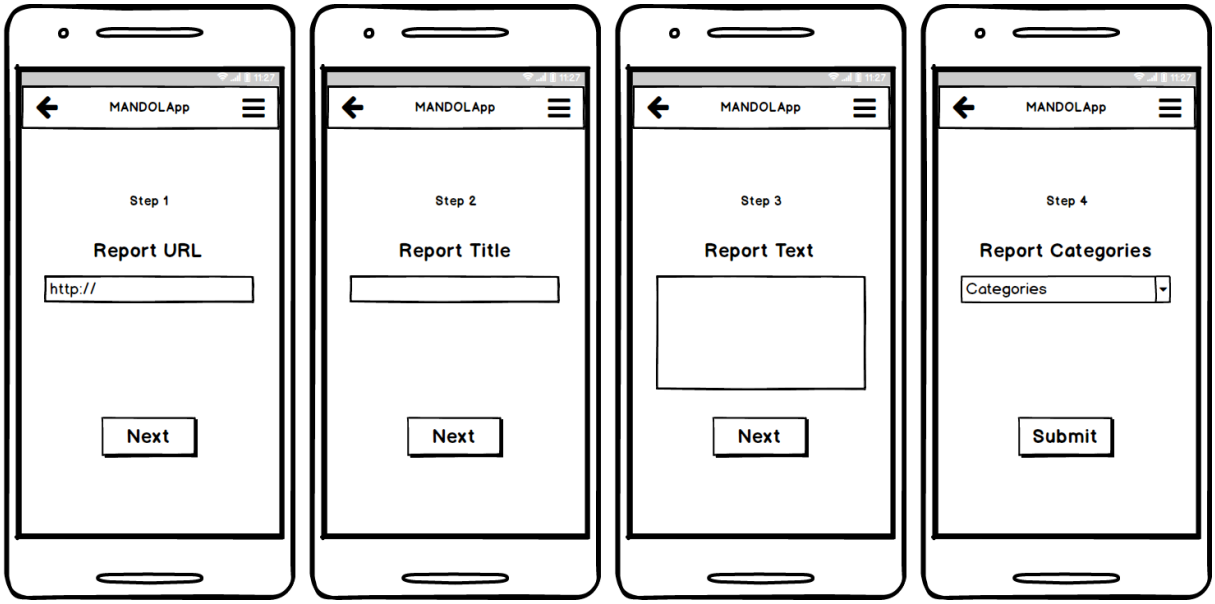

Figure 5: Reporting methodology steps mock-ups.

4 Development

In the development phase, all of the above design concepts and mock-ups are taken into account in order to deliver the mobile application. In this section the development of Smartphone app will be presented, including the technologies, tools and frameworks which have been used.

4.1 Cordova: Multiple Platforms with One Code Base

The smartphone app will be compatible with Android, iOS and Windows platforms. The application has been developed using the Apache Cordova framework [10]. Apache Cordova is a popular mobile application development framework which enables wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid, meaning that they are neither truly native mobile application nor purely Web-based.

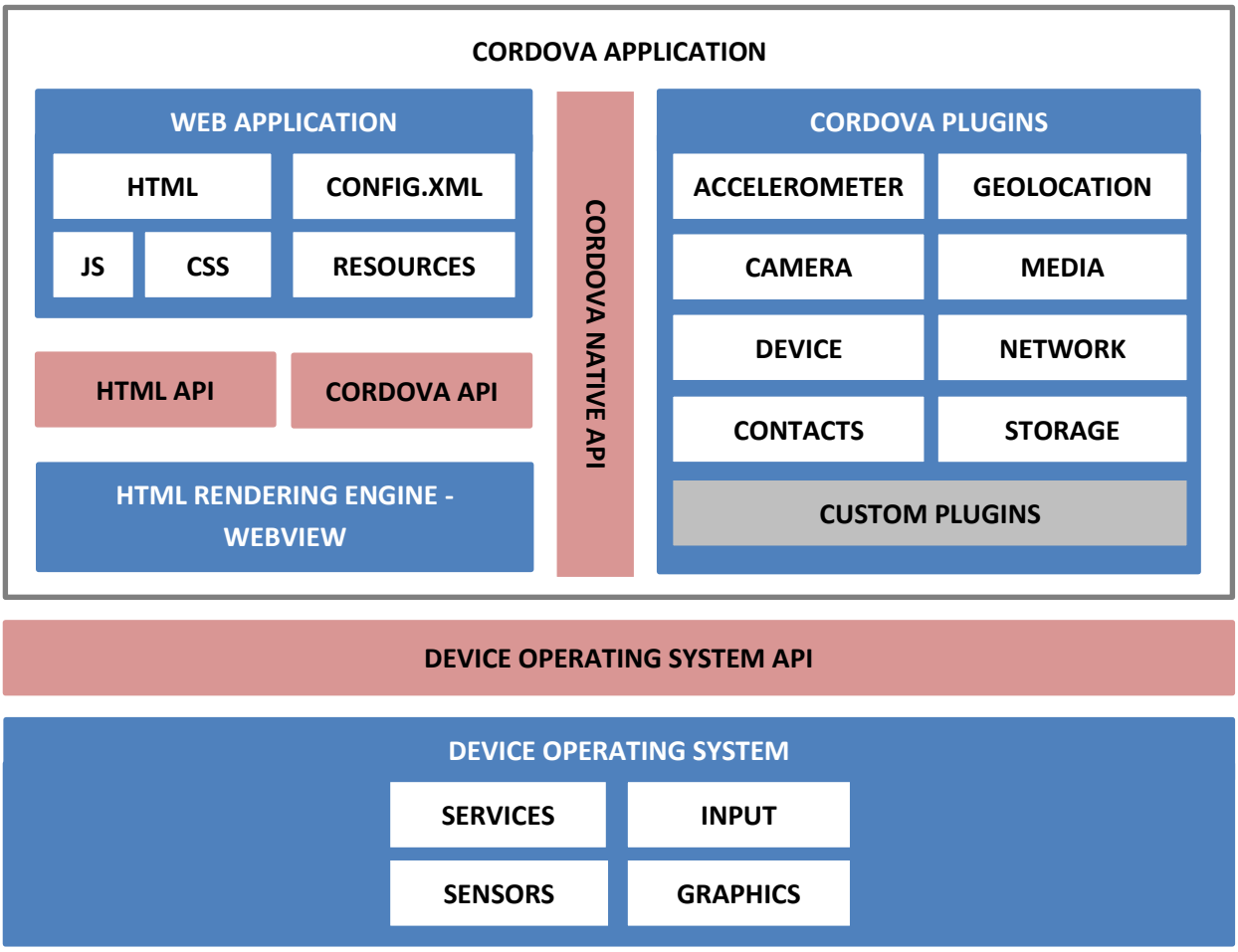

Figure 6: Cordova development framework architectural diagram

Apache Cordova also provides the ability of creating plugins. A plugin is a package of injected code that allows the Cordova web-view within which the app renders to communicate with the native platform on which it runs. Plugins provide access to device and platform functionality that is ordinarily unavailable to web-based apps. So, by using such plugins, the

core application can be provided with any native functionality, based on the platform. Figure 6 depicts the architectural diagram of the Cordova development framework.

During the smartphone app's development, a lot of those plugins were used and created in order to gain this native functionality for several of the application's modules. Table 1 below is showing the correspondence of the different mobile modules with their respective plugins.

Module	Plugin	Status	Description
MANDOLA Bubble	com.ab.cordovafloatingactivity 0.7.0	EDITED	Responsible of the MANDOLA Bubble service.
MANDOLA Bubble	cordova-plugin-background-mode 0.6.6	EXISTED	Responsible for running in background mode
MANDOLA Optical Character Recognition	cordova-plugin-camera 2.3.2	EXISTED	Responsible for accessing the user's gallery to select screenshot
MANDOLA Optical Character Recognition	cordova-plugin-crop 0.3.1	EXISTED	Responsible for cropping the selected screenshot
MANDOLA Optical Character Recognition	cordova-plugin-my-tess-two 0.1.0	EDITED	Responsible for recognizing the cropped text
MANDOLA Optical Character Recognition	cordova-plugin-file 4.3.1	EXISTED	Responsible for accessing tesseract and cropped image files
MANDOLA Optical Character Recognition	cordova-plugin-file-transfer 1.6.1	EXISTED	Responsible for managing tesseract and cropped image files
MANDOLA Proxy	cordova-plugin-inappbrowser 1.6.0	EXISTED	Responsible for loading browser within the application
MANDOLA Reporting	cordova-sqlite-storage 1.5.2	EXISTED	Responsible for managing the local memory of user reports

All	cordova-plugin-crosswalk-webview 2.2.0	EXISTED	Providing runtime with Chromium and Blink of Google

Table 1: Cordova plugins used by Smartphone app.

4.2 Reporting Modules Development

In this section the focus is on the development analysis of the reporting modules. Both of the reporting mechanisms will be explained in detail about how they work and which of Apache Cordova plugins they use. Also, the development of the MANDOLA Bubble will be analyzed.

4.2.1 Optical Character Recognition – OCR Module

The Optical Character Recognition module utilizes the Tesseract OCR system [11]. The Tesseract OCR system is available under the Apache license and its development was sponsored by Google. It is also considered as one of the most accurate open-source OCR engines that are currently available. It can recognize a total of 100 languages, including English, Greek, French, Spanish and Bulgarian. It is available for Linux, Windows and Mac OS X but there are several implementations of it in order to run on mobile devices. The smartphone app uses the Apache Cordova plugin that supports OCR using Tesseract, which did undergo several changes in order to serve its needs. The diagram of Smartphone app OCR module is depicted in Figure 7.

The optical character recognition module receives as input the cropped image that contains hate speech, as well as the language of the content. To do so, the user goes through the process of selecting the screenshot from within his/her device gallery and loads the selected image in a cropper where the image must be cropped in the text boundaries. After the cropping is done, the new image along with the selected language goes in the Tesseract plugin. The selected language can be downloaded and configured from the application settings (Section 5.5). Then the Tesseract loads the analogous language file and initiates the character recognition process. After the recognition is done, the generated text is received from the core application and loaded into the hate speech report.

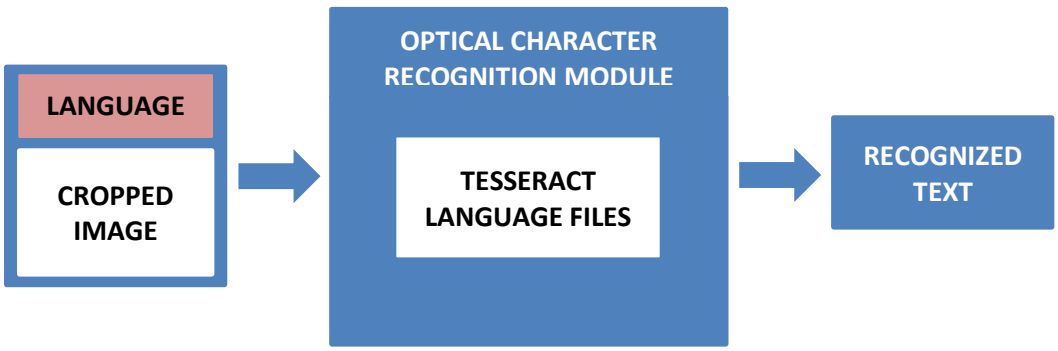

Figure 7: Smartphone app OCR module diagram.

4.2.2 MANDOLA Proxy Module

The MANDOLA Proxy method, as stated earlier, is for any public encounter of online hate speech. By public, we mean an encounter in web sites or social media where no credentials are necessary in order to view the content of the URL, e.g. a YouTube video link, a public Facebook group, a web site etc. The MANDOLA Proxy utilizes the Wayback Machine, which is a digital archive of the World Wide Web. By 2015, the Wayback Machine had archived 452 billion web pages over time and is still growing at a rate of about 20 terabytes each week. This service enables users to see archived versions of web pages across time. The implementation of the MANDOLA Proxy was done with PyWb, a Python implementation of the Wayback Machine. The diagram of Smartphone app Proxy module is depicted in Figure 8.

The MANDOLA Proxy uses the Wayback Machine in order to gain the ability to append the reporting code within the web site and to prevent the user from logging in any web site that needs credentials. Also, it uses the TextHighlighter, a JavaScript library that enables website text highlight. The user can load the URL via the MANDOLA Proxy and highlight the hate related content. After the text is highlighted, the reporting form is shown in order to be filled by the user.

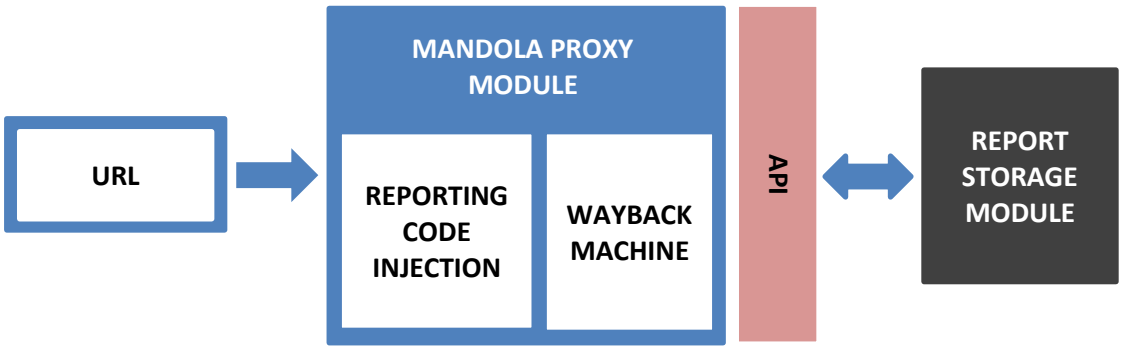

Figure 8: Smartphone app Proxy module diagram.

4.2.3 MANDOLA Bubble Module

To provide users with a simpler and faster way of reporting, the MANDOLA Bubble was implemented and added as a background service to the Smartphone app. The diagram of Smartphone app Bubble module is depicted in Figure 9. MANDOLA Bubble is a background process that the user can enable from the settings view. It is built as a Cordova Plugin and utilizes the existing background mode plugin. As soon as the Smartphone app is set to the background, the MANDOLA Bubble service is triggered and listens to possible URL copy events. A URL copy event is considered as any URL copy action done by the user from a source such as a mobile browser, or any native social media applications (e.g. Twitter, Facebook). The MANDOLA logo is drawn on the device's screen which is used to display any sensed URL copy event, and by pressing it, the user will generate the MANDOLA report for that URL. The on-screen drawing of the MANDOLA logo is only supported for Android devices. For IOS devices the MANDOLA Bubble is still functional but instead of using the on-screen drawing, it utilizes

the local notification services. If the user encounters hate speech while browsing any native social media application, he/she can copy the URL of the post and the service will capture this. Then, the user will be notified and by pressing the MANDOLA Bubble, the copied URL's report will be automatically created and filled without needing to swap between processes. Afterwards, the user needs to validate the fields and select the corresponding hate categories. The use of this feature might encourage users to actively use the smartphone app.

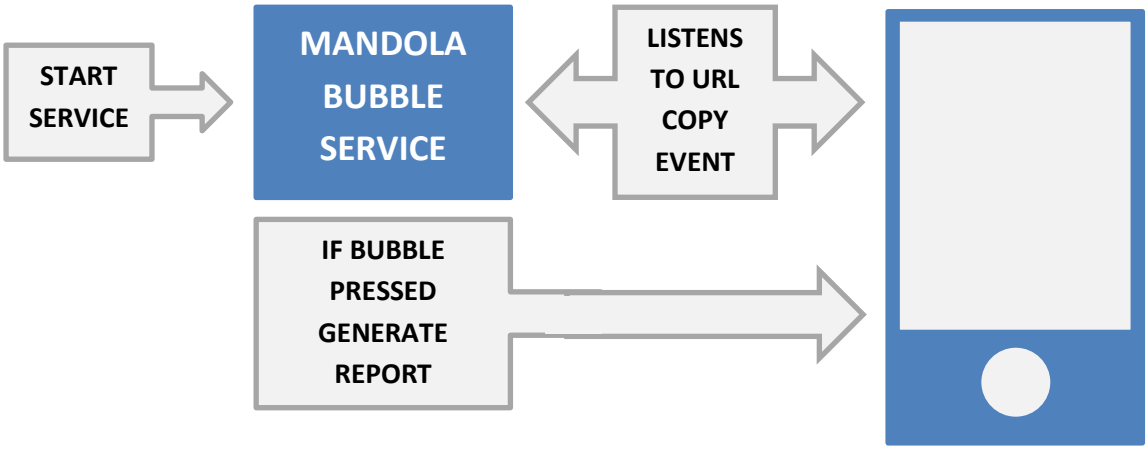

Figure 9: Smartphone app Bubble module diagram.

5 Interfaces

The Smartphone app's interfaces were implemented based on the above analysis and design. Utilizing Apache Cordova, the procedure of implementing the interfaces is the same as building a web application, using HTML5, CSS3 and JavaScript. The whole application is based on the MVC-Model View Controller design pattern, where there is a Controller JavaScript application which changes the Views based on the input of the user.

5.1 Awareness View

The objective of the Awareness View is to raise the user's awareness on the problem of online hate speech, and also to provide him/her with information that shows the impact of hate speech to the world. This is done by providing visualizations of survey results, analysis on events and MANDOLA Monitoring Dashboard's statistics. There are several charts presented, about global and local analytics on online hate speech. Some of information is static, taken from surveys [12], and some is dynamic using the MANDOLA Monitoring Dashboard's API. Figure 10 and Figure 11 depict the static information provided to users through the application.

Figure 10: Awareness View – Hate speech encounters and Category statistics.

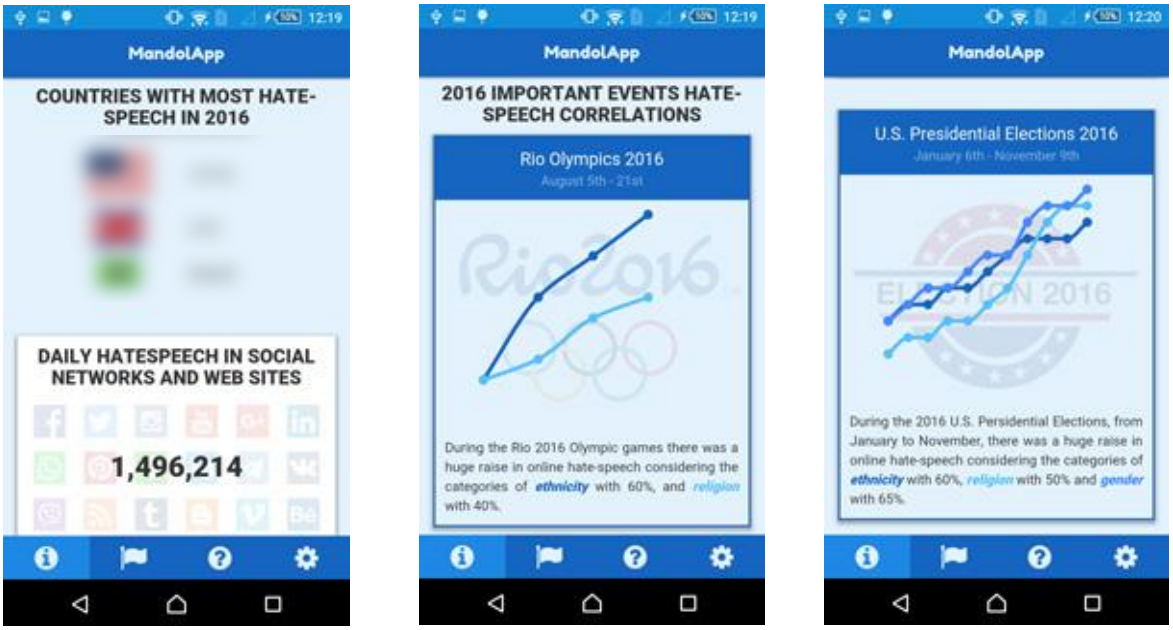

Figure 11: Awareness View – Event specific hate speech statistical analysis.

5.2 FAQs View

The FAQs View aims to support the Awareness View on the matter of online hate speech, but also to provide users with answers to the most common questions that concern hate speech, online hate speech and legal matters of it. This View (as depicted in Figure 12) is based on the MANDOLA Project 4.1. Deliverable and it provides a more compressed version of the answers.

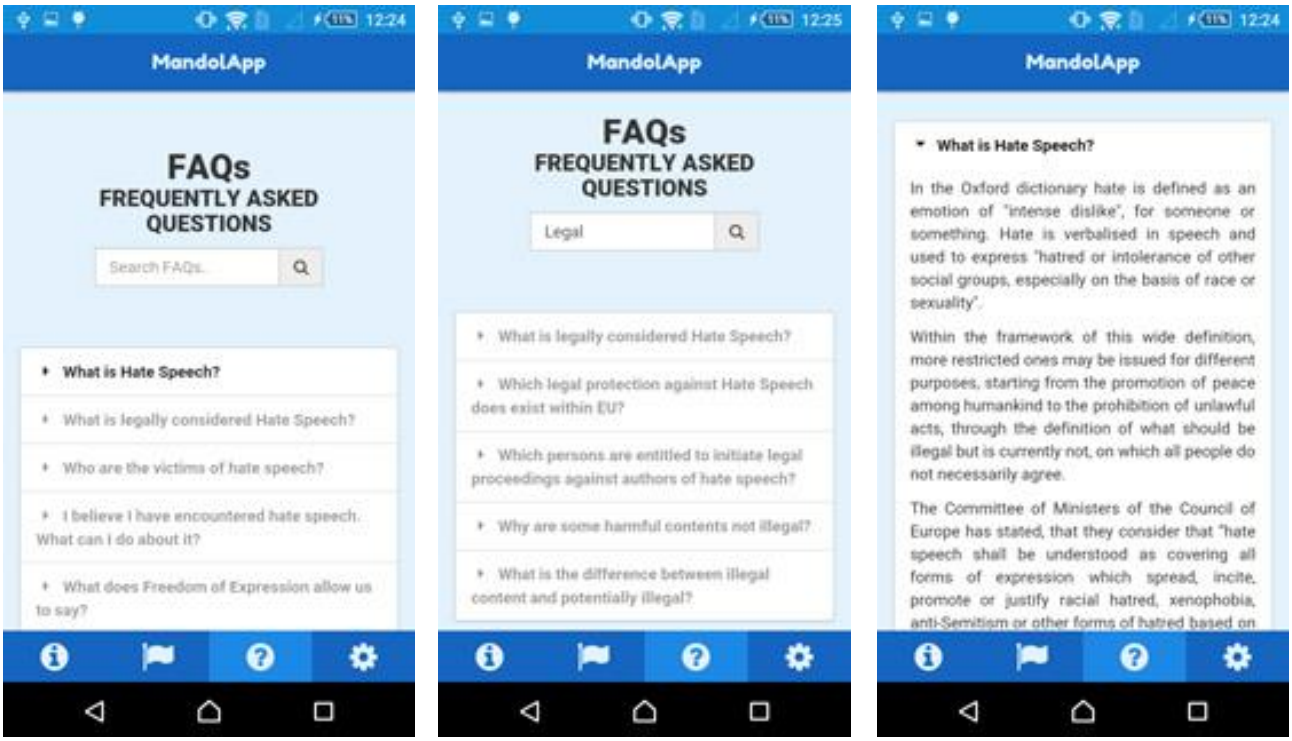

Figure 12: FAQs View screenshot.

The user can find all the hate speech's legal aspect related FAQs based on the 4.1 Deliverable of the MANDOLA Project. The answers are presented in smaller 'knowledge chunks', where the full ones can be found in the actual deliverable. The user can filter the questions based on specific keywords he/she wants to search for and reduce the number of the FAQs. By pressing a button, the clicked question's accordion will drop and the answer will be visible to the user. By clicking the accordion again, the answer will close.

5.3 Report View

The Report View (depicted in Figure 13 and Figure 14) is considered as the core of the smartphone app. It consists of the reporting mechanisms and the previous list of reports. It is a simple list view with a floating button to create new reports. Both the OCR and MANDOLA Proxy methods use the main reporting view, where the previous reports by the user are presented. To manually create a new report, the user must press the button and the two options will show. The button is used in order to create a report for public content via the MANDOLA Proxy, where the button is to utilize OCR Module for private content.

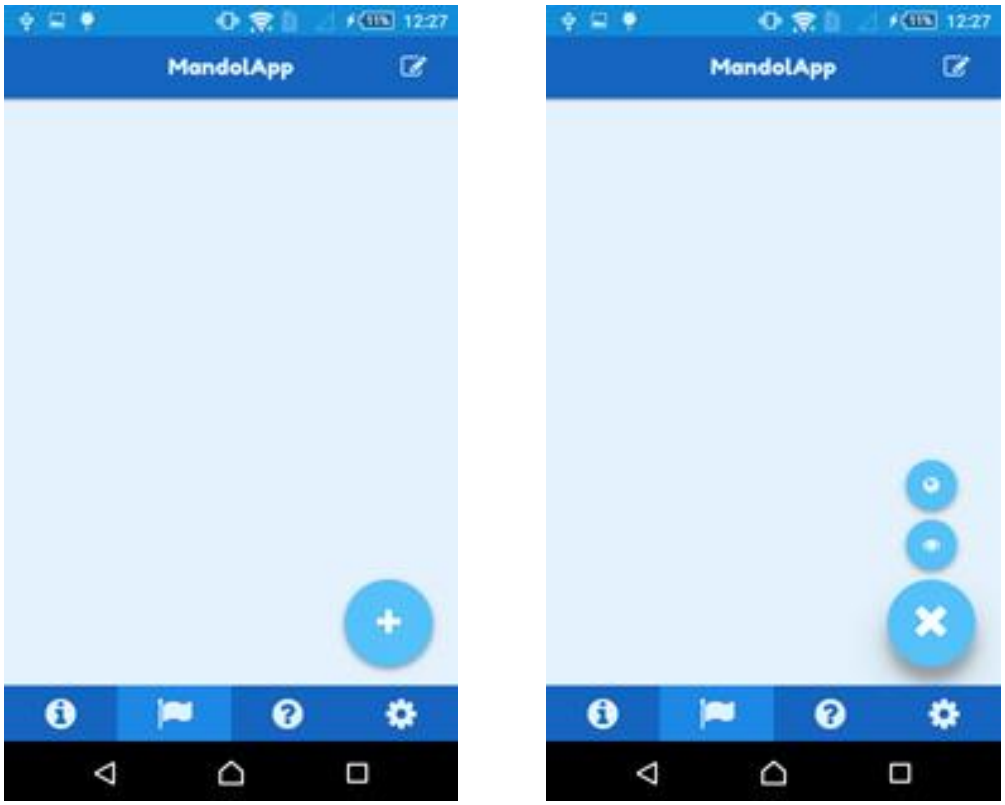

Figure 13: Report View – Disabled/Enabled "Create" floating button.

5.3.1 Report using MANDOLA Proxy

If the user selects the MANDOLA Proxy reporting method, then he/she will be prompted to write the hate speech containing URL. By pressing "Load" the URL will be loaded in the InAppBrowser via the MANDOLA Proxy Server.

Figure 14: Report View / MANDOLA Proxy – Create report form, URL field.

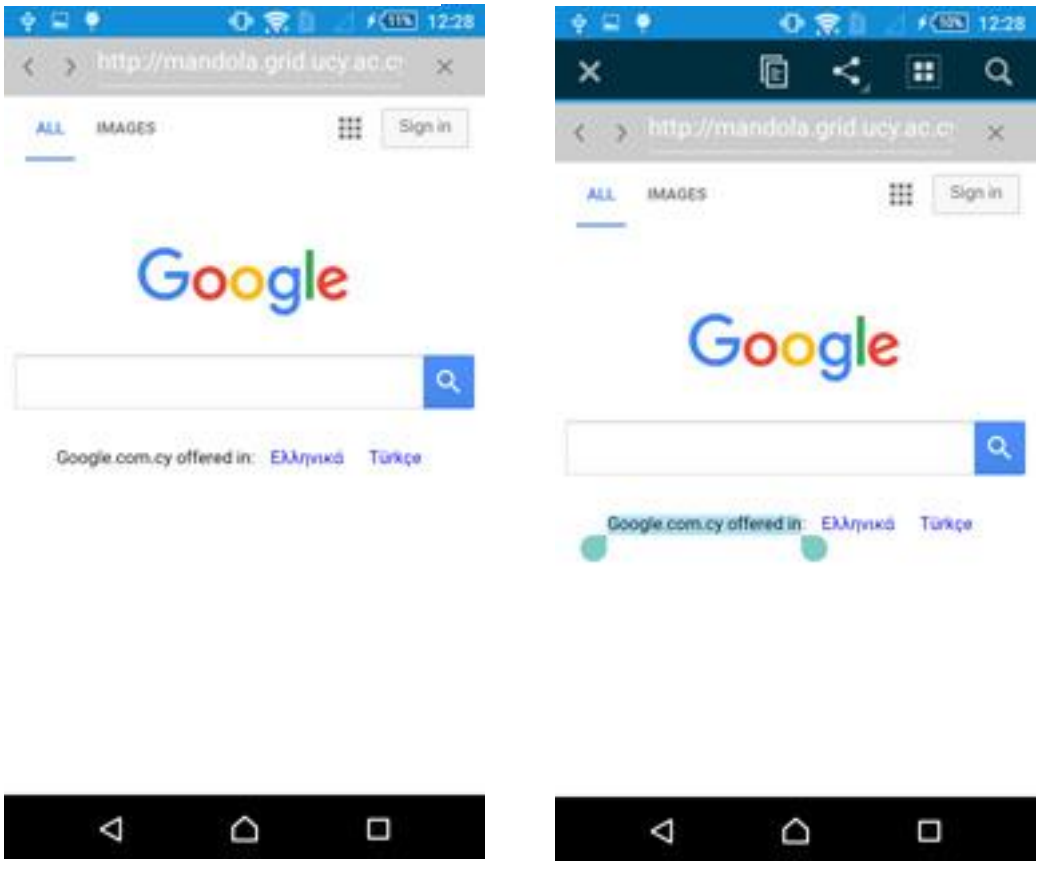

Figure 15: Report View/ MANDOLA Proxy – InAppBrowser loaded site and highlighted text.

Afterwards, the user can highlight the hate speech within the URL and by pressing the highlighted text, the MANDOLA Reporting form will show. This functionality can be seen in Figure 15.

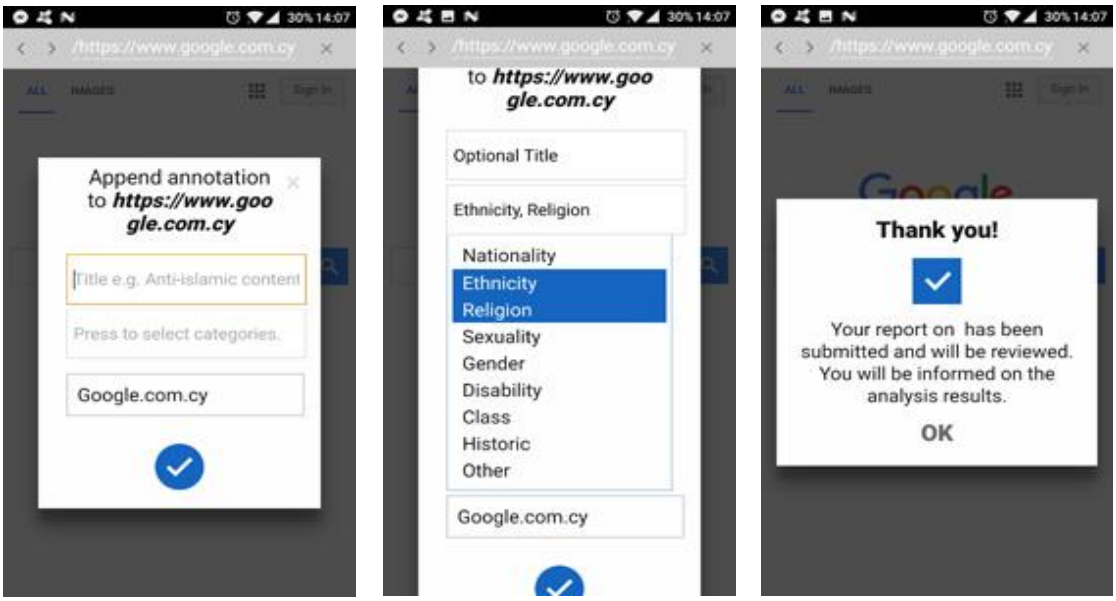

Figure 16: Report View/ MANDOLA Proxy – InAppBrowser MANDOLA report.

The MANDOLA Reporting form has the same fields as the OCR Form of the Smartphone app. First the user can optionally fill the title field for describing further the report. Then, the user needs to select the appropriate hate categories that characterize the reported text. After filling the form, the user submits the form and receives a success message. If something goes wrong, an error message will be shown.

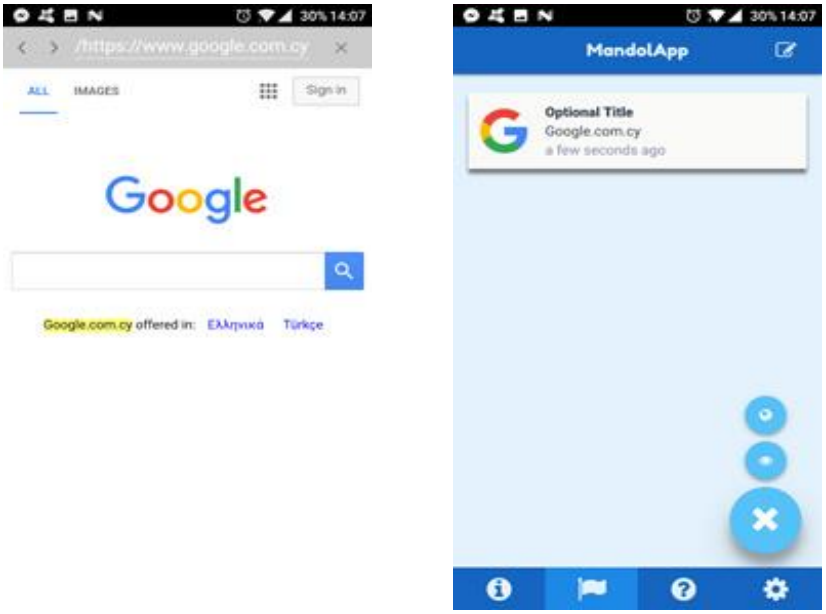

Figure 17: Report View/ MANDOLA Proxy – MANDOLA Reported text highlighted and report item appended.

The reported text can be seen as highlighted from within the InAppBrowser via the MANDOLA Proxy (Figure 17). If the user re-opens the URL, then the reported text will be highlighted. When the user closes the InAppBrowser, he/she can see the newly reported item in the Report View.

5.3.2 Report using OCR

When the Optical Character Recognition Reporting method is selected, the user will be prompted to load the screenshot from the device's gallery. After the image is selected, it will be loaded in the cropping system in order to crop the hate containing text for the Tesseract to recognize it. The cropping system is the native one provided by the Apache Cordova plugin called *cordova-plugin-crop 0.3.1* and can be seen below in Figure 18.

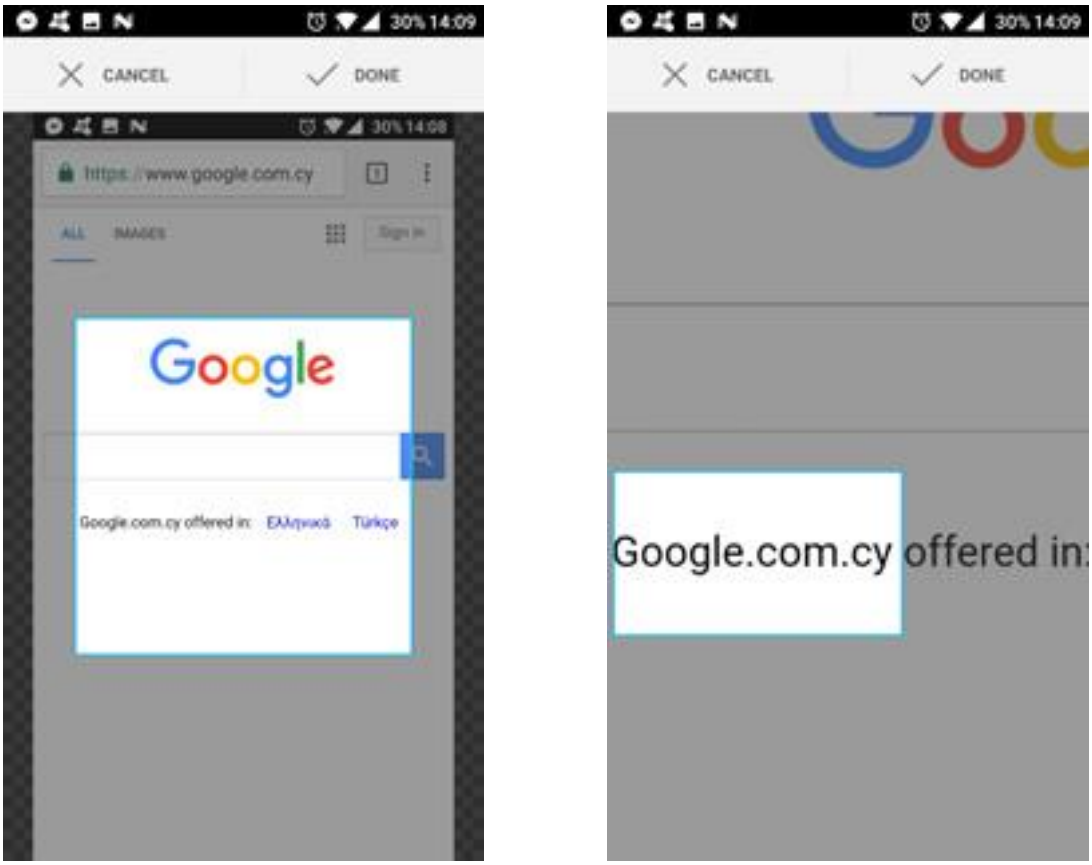

Figure 18: Report View/ MANDOLA OCR – Screenshot loaded in Cordova's crop plugin.

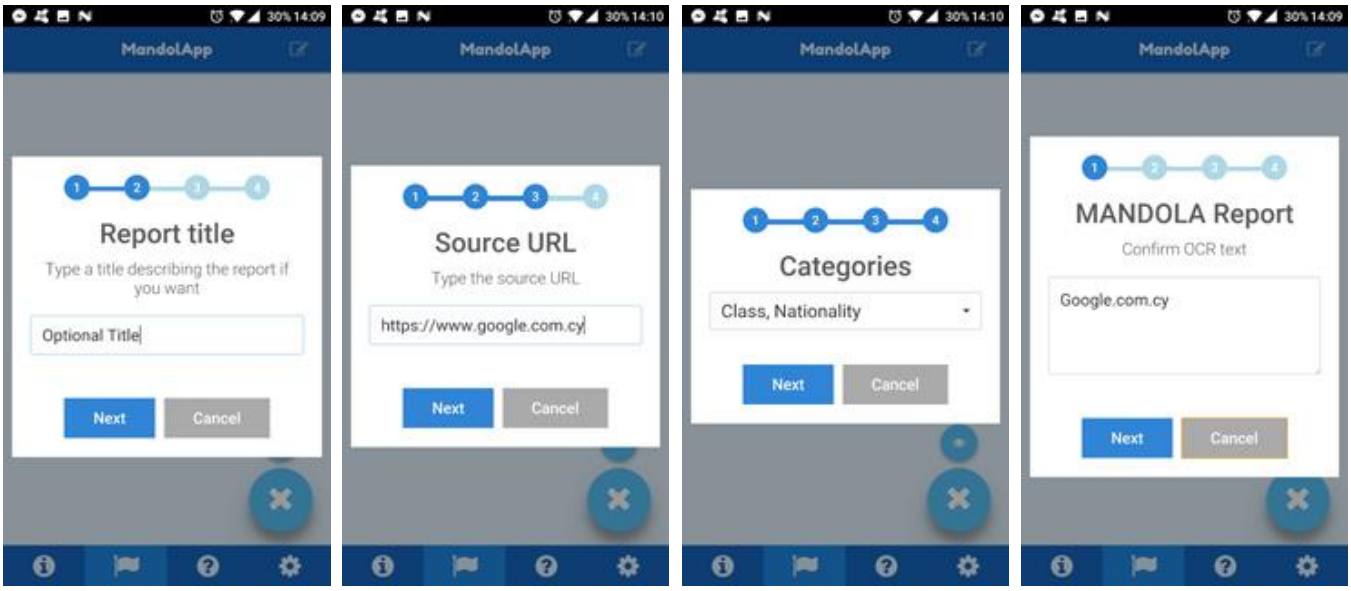

Figure 19: Report View/ MANDOLA OCR – MANDOLA Reporting steps.

After completing the cropping of the screenshot, then Tesseract recognizes the text and loads it in the MANDOLA Reporting form (Figure 19). Then the user proceeds in providing the information of the remaining fields. After completing the form filling in the four simple steps, then the user presses "Next" which will submit the report into the system. The newly report item then is shown in the Report View, just like the MANDOLA Proxy method.

5.4 MANDOLA Bubble View

Figure 20: MANDOLA Bubble enabled.

By enabling the MANDOLA Bubble from within the settings view, it can be visible when the smartphone app goes to background mode.

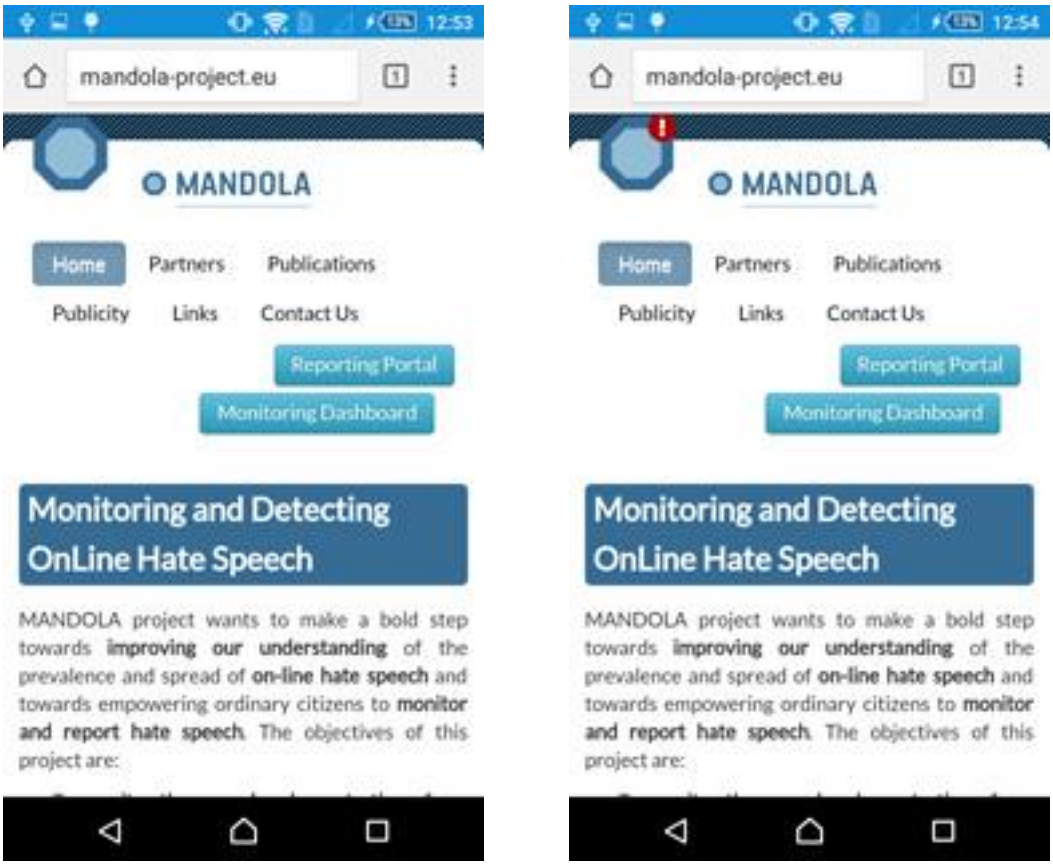

Figure 21: MANDOLA Bubble sensed URL copy event.

When the user copies a URL from the mobile browser or a native social media application, then the MANDOLA Bubble shows an exclamation mark meaning that it sensed a URL Copy Event (Figure 21).

Figure 22: MANDOLA Bubble method prompt message.

When the user presses the MANDOLA Bubble with the exclamation mark, then he is prompted to choose the Reporting method, like in Figure 22. Afterwards, the user only needs to highlight or crop the hate containing text.

5.5 Settings View

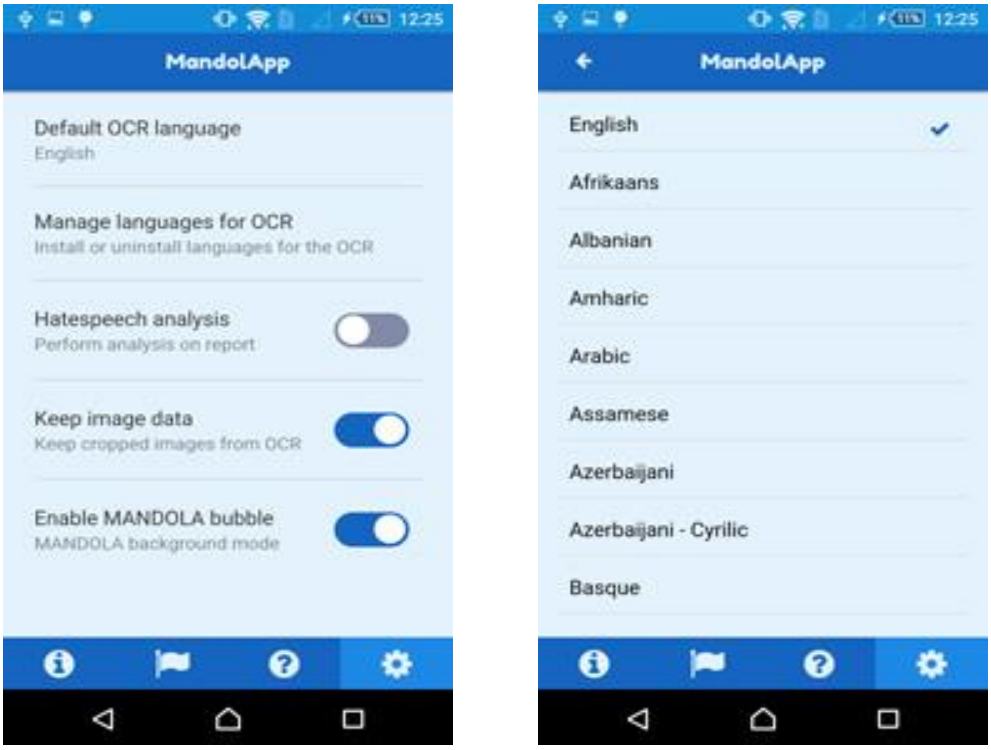

Figure 23: Smartphone app Settings View.

By pressing the "Manage languages for OCR" option, a list with all the supported languages of Google's Tesseract are loaded (Figure 23). The user can press any of these languages in order to download them and configure OCR to recognize them. The downloaded languages have a ✓ symbol to their right and by pressing them, the user is prompted to delete them.

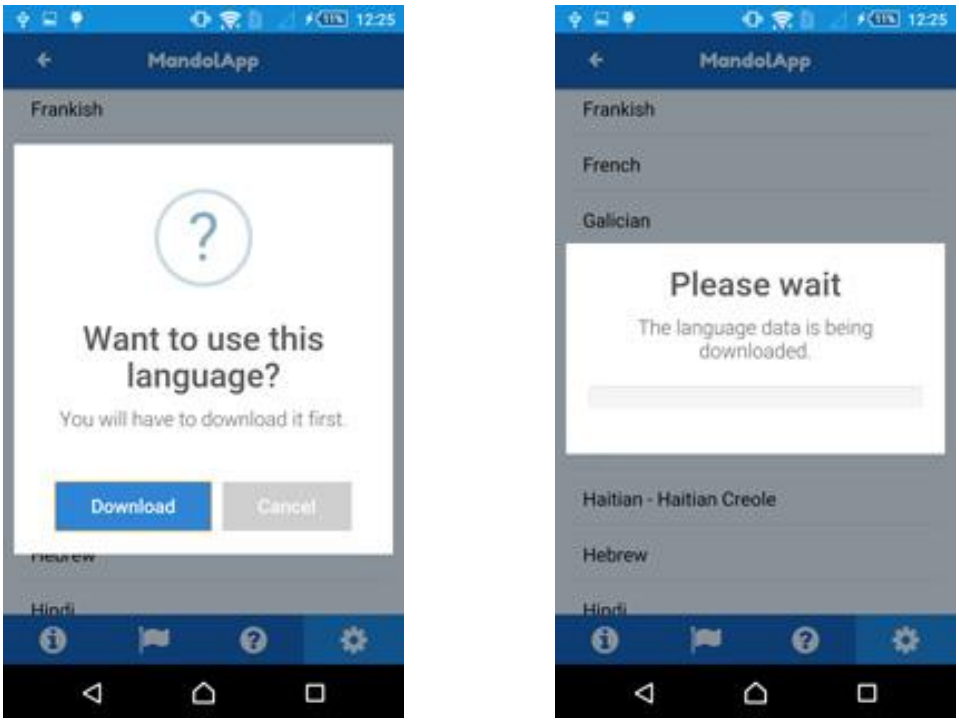

Figure 24: MANDOLA OCR language data download message.

When a language is pressed, the user is prompted to download it or cancel. If the “Download” button is pressed, then a loading bar is shown and presents the download progress of the language data, depicted in Figure 24. The downloaded languages can be found in the “Default OCR Language” option in settings view. The user can select the default OCR language from there. After a new default language is set successfully, the appropriate message will show (Figure 25).

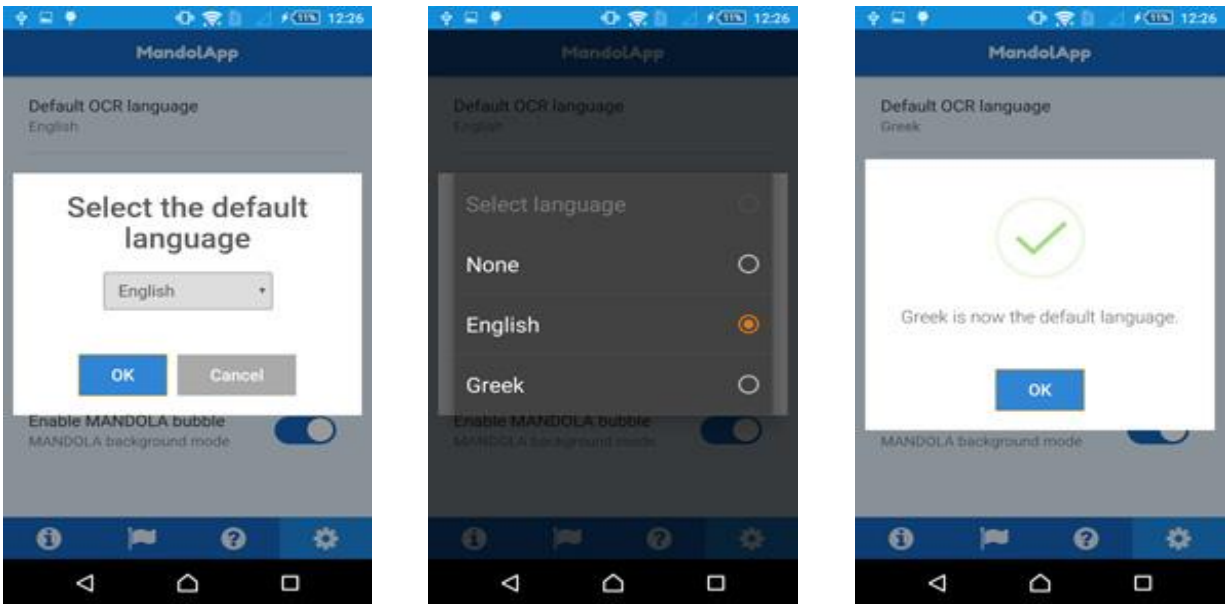

Figure 25: MANDOLA OCR default language setup.

6 Data Management

In this section, the backend that communicates with the Smartphone app will be explained and analysed. The backend, as depicted in Figure 26, consists of three sub-modules that are responsible for processing and storing the MANDOLA reports: the API, the Storage Module and the Hate Analysis Module.

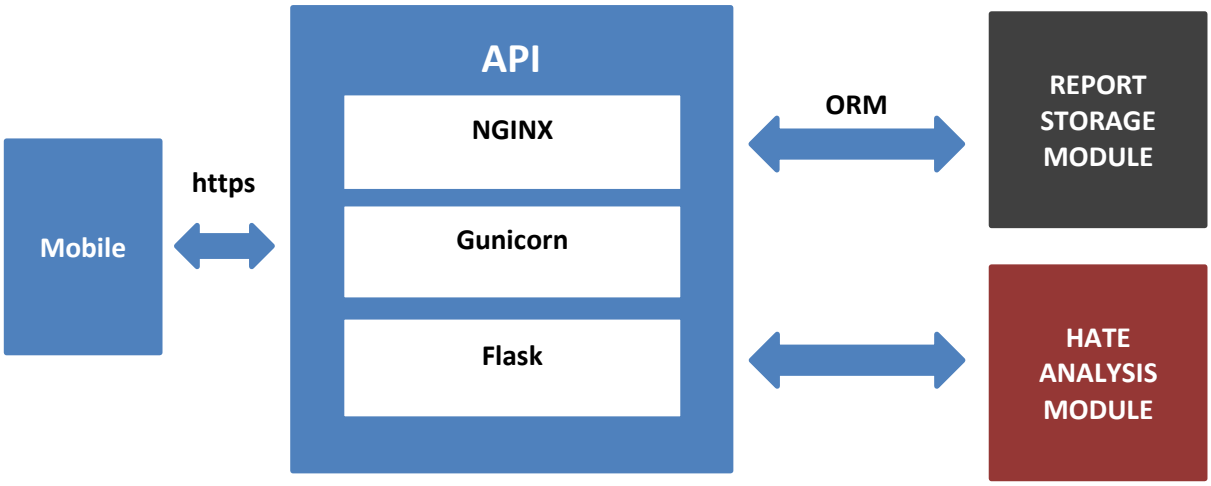

Figure 26: Data Management architectural diagram.

6.1 API

The Smartphone app uses an API to store the reports and also to check if a report is considered hate speech using the Hate Analysis Module. Furthermore, the API can be used by any authorized 3rd party organization in order to retrieve reports. The *RESTful API* was developed using a 3-layer server architecture and allows only secure https requests (Figure 27).

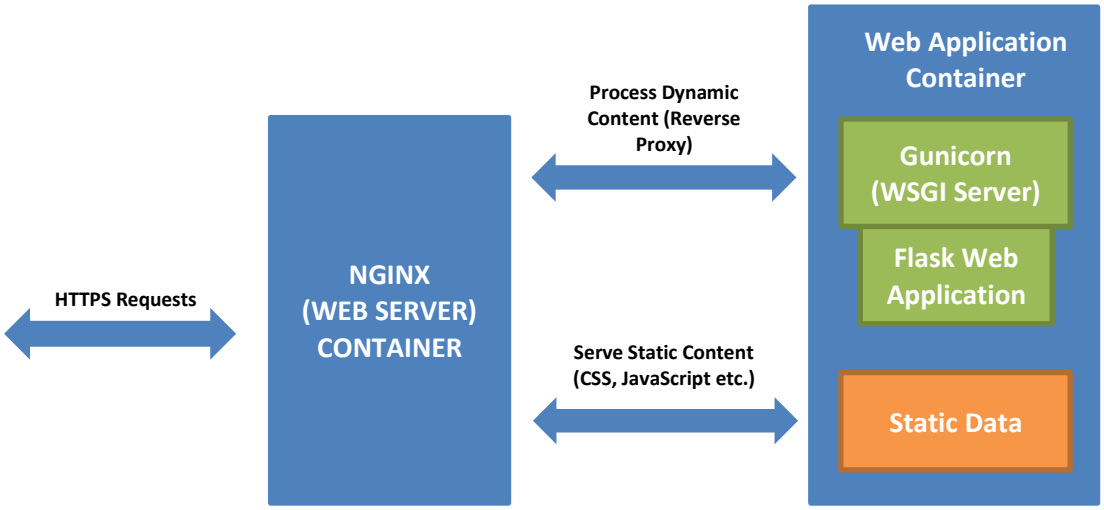

Figure 27: MANDOLA Reporting App API.

The first layer consists of the main web server called NGINX [13]. NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server with a small memory footprint. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.

Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Furthermore, NGINX scales in all directions: from the smallest VPS all the way up to large clusters of servers.

The second layer consists of the WSGI (Web Server Gateway Interface) HTTP Server middleware called Gunicorn [14]. Gunicorn is a pre-fork worker model ported from Ruby's Unicorn project and it is simply implemented, light on server resources, and fairly fast.

The WSGI middleware has two sides: the "server" or "gateway" side, in our case the web server Nginx, and the "framework" side, Flask. To process a WSGI request, the server side utilizes the Flask framework and provides environment information and a callback function to the framework side. The framework processes the request, returning the response to the server via the middleware using the callback function that it was provided.

The WSGI middleware implements both sides of the API. The server receives a request from a client and forwards it to the middleware. After processing, it sends a request to the framework. The framework's response is forwarded by the middleware to the server and ultimately to the client. So, it basically makes a communication channel between NGINX and the Flask framework.

The third layer of the server architecture consists of the Flask framework [15]. Flask is a micro framework written in Python which does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. The framework is responsible for providing the API functionality and also communicating with the Storage Module and the Hate Analysis Module.

Each layer is needed in order to achieve a fast and lightweight backend system. Also, each layer can be hosted on separate machines for better load-balancing. Without Gunicorn there would be no way of communication between Flask Application and NGINX. Also, without a proxy (NGINX) buffering slow clients, the Gunicorn would be easily susceptible to denial-of-service attacks.

MANDOLA Reporting App API main functionalities are listed below:

- Storing the reports.
- Retrieving the reports.
- Analyzing a new report using the Hate Analysis Module.

Resource	Type	Example	Description
/api/new-report	POST	text="this text includes hate speech."	Sends a new report.

		url=www.hatespeech.com country=Cyprus date=2017-02-10 time=12:00 tags=ethnicity,nationality,religion	
/api/reports/country	GET	/api/reports/country?from=2010-09-14&to=2014-12 04&country=United%20Kingdom	Returns the reports from a specific country in a specific date range.
/api/analyze-text	POST	text=”this text does not include hate speech.”	Sends a text and returns the probability of the text to be hate speech, but also the possible categories.

Table 2: Smartphone app API Resources.

6.2 Report Storage Module

The Report Storage Module is developed for testing purposes but still takes into consideration data privacy policies by protecting the user’s identification from the information stored. Thus, no sensitive data is stored in the database. In deployment, the reports will go directly to the reporting authorities via the API.

6.2.1 Object-Relational Mapping

As mentioned above, the Flask framework is communicating with the Report Storage Module. The communication is done by using an extra layer called ORM - Object-Relational Mapping between the database and the framework. The ORM is useful for speeding development by eliminating repetitive code and making data access more abstract and portable. The ORM that has been used is called Peewee. Peewee is a simple, lightweight and expressive ORM that provides built-in support for MySQL, the re-use of query fragments and the safety against SQL injections.

6.2.2 Database

The data are stored in MySQL database, which is an open-source well documented relational database management system RDBMS which provides strong data protection and high performance. Furthermore**,** all the reports that’s inside the database are encrypted using a python library called **cryptography.** The Smartphone app data structure is presented in Table 3.

Property	**Description**

text	The text that might contain hate speech.
url	The url that might contain hate speech.
lang	The language in which the hate speech is written.
country	The country that the report was originated from.
date	The date that the specific report was created.
time	The time that the specific report was created.
tags	An array of the hate categories. The probable categories are ethnicity, nationality, gender, religion, sexual, class, disability, politics, and sports.

Table 3: Smartphone app database structure.

6.3 Hate Analysis Module

The Hate Analysis Module is responsible for determining whether a text provided as input is considered hate-speech and to which categories it can be classified into (e.g. religion, nationality, ethnicity etc.). Basically, it is the module that gives insight to the user about a possible MANDOLA report.

In order to recognize hate speech, a text classifier was trained with a corpus consisting of annotated hate and not hate speech text. The corpus used is the one collected and enriched from the social scientists. The corpus is in several languages from the Member States and thus the classifier is language agnostic and performs multi-lingual classification. Furthermore, considering that a potential hate-related input can be labeled with different hate categories, such as religion, nationality and ethnicity, the classifier also supports multi-label classification, with the hate categories being the labels. Details can be found in Deliverable D3.1.

The user can enable the Hate Speech Analysis from the Smartphone app's settings view. By doing so each time the user creates a new MANDOLA Report, the possible hate speech content will go through the Hate Speech Analysis module and provide the user with the classifier's output in order to provide him/her with an insight.

7 Use Cases & Metrics

The implementation of the Smartphone app is focused on the usability and the user experience. Through the selected reporting methods and with the MANDOLA Bubble, the Smartphone app has eliminated flaws of other applications and targets the today's user where he/she moves in a fast pace. In this section, there will be shown and analysed use cases of online hate speech encounters and reports.

7.1 Capturing URLs in Social Media Applications

Most of the online social network native applications give the functionality to user, to copy a link to a certain post in order to share it or send it. The Smartphone app utilizes this functionality in order to provide the URL of the hate speech containing post, in the MANDOLA Report. In this subsection, we provide examples of this functionality for Facebook and Twitter native mobile applications.

For Facebook, each post that the user sees in his/her timeline has a chevron at the top right corner. By pressing that chevron, options about the specific post are shown and one of these options is the Copy Link. By pressing that the URL pointing to the specific post copied to the device's clipboard (Figure 28). The MANDOLA Bubble would have sensed the URL Copy Event and the exclamation mark would be shown.

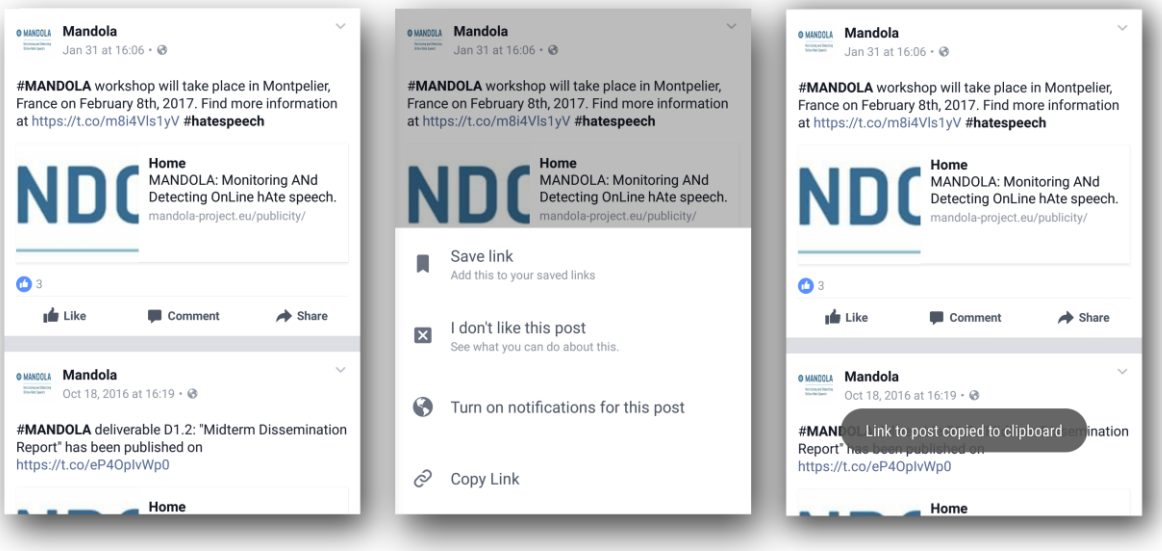

Figure 28: Facebook mobile application steps to copy post's URL.

The same is done for the Twitter native application (Figure 29). While the user browses his/her feed, can see the same chevron at the top right corner of each tweet. By pressing the chevron the tweet's options are shown and, as with Facebook, the option "Copy link to Tweet" can copy the URL to the specified tweet, which would again be sensed by the MANDOLA Bubble.

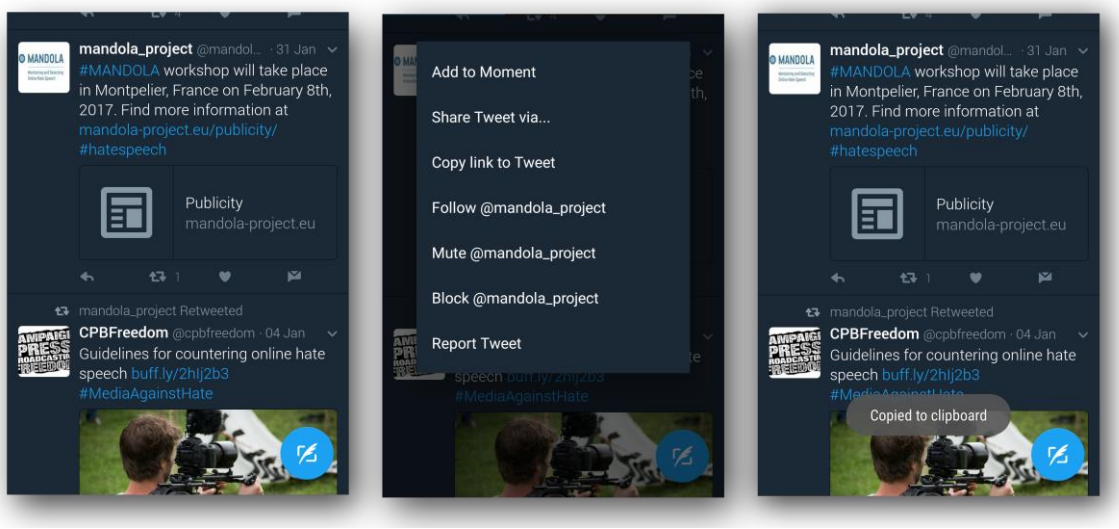

Figure 29: Twitter mobile application steps to copy post's URL.

7.2 Public Hate Speech Encounter

7.2.1 Using a Mobile Browser

When a user encounters public hate speech while using his/her mobile web browser, considering that the MANDOLA Bubble is enabled, he/she can simply copy the URL from the browser's address bar, and the MANDOLA Bubble will sense it (Figure 30). As a result, the red exclamation mark will be displayed, and when the user presses the bubble, the MANDOLA report will be generated automatically. Then the user will select the MANDOLA Proxy method, where the URL will load to highlight the hate speech containing text.

Figure 30: MANDOLA Reporting workflow using mobile browser.er.

7.2.2 Using Twitter Native Application

As shown before, the twitter native application provides the functionality to copy the link for a specific tweet. By applying the same process, while having the MANDOLA Bubble enabled, the user can report any public encounter of hate speech in the twitter application. Simply by copying the link via the "Copy link to tweet" option, the MANDOLA Bubble senses it and the exclamation mark appears (Figure 31). Just like any other URL Copy sensed event, by pressing the bubble, the MANDOLA Report is automatically generated.

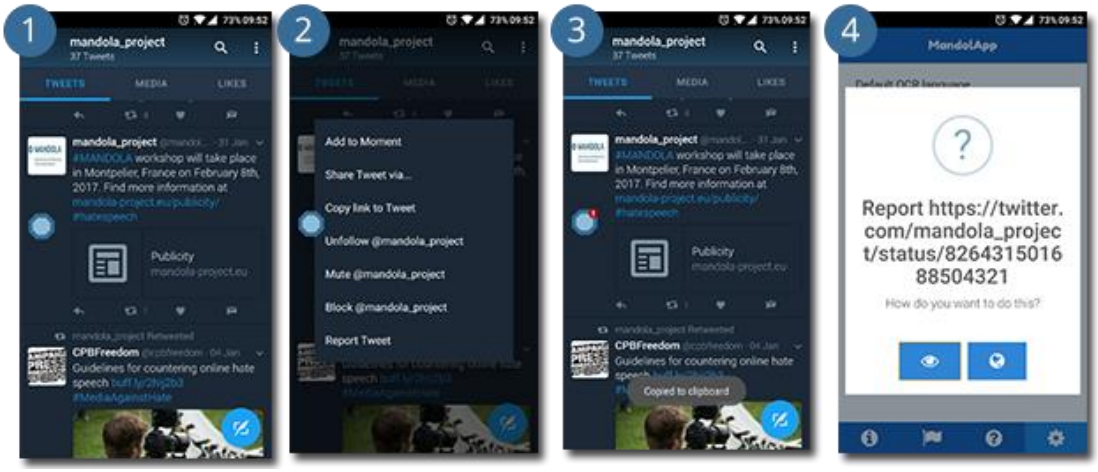

Figure 31: MANDOLA Reporting workflow using mobile Twitter application.

7.2.3 Using Facebook Native Application

With the same work-flow as with the mobile browser and the native twitter application, the user can generate the MANDOLA Report in Facebook native application. By pressing the "Copy Link" option in a specific post, he/she can press the MANDOLA Bubble with the red exclamation mark and generate automatically the MANDOLA Report for the specific hate speech containing post (Figure 32). The user then proceeds with selecting the MANDOLA Proxy method and after InAppBrowser loads the site, he/she highlights the hate speech containing text.

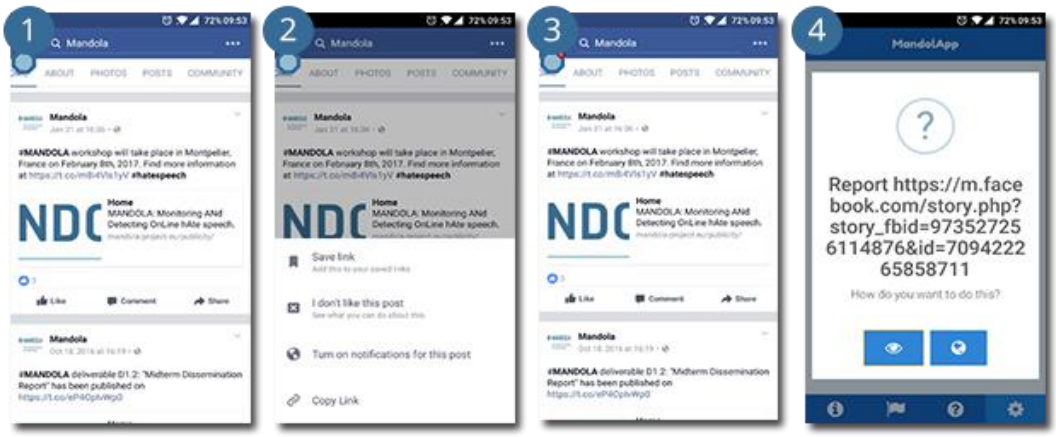

Figure 32: MANDOLA Reporting workflow using mobile Facebook application.

7.3 Private Hate Speech Encounter

While public hate speech encounters require only the URL to be copied, private hate speech encounters require also a screenshot of the hate speech containing post or comment. Then the OCR module is used to recognize the text and generate the MANDOLA Report.

7.4 Metrics

In order to evaluate beforehand the Smartphone app, so as to proceed in submitting it to the various Mobile Application Markets, a sample usage metrics analysis has been done. The application was tested in Android versions 5.1.1, 6.0 and 7.0 on actual devices. It was also tested on iOS versions 10.0, 10.1 and 10.2 via emulators. The analysis was done using

Chrome's debugging tools. This is possible by enabling "USB remote debugger" within our android device and plug with a USB cable. Chrome will detect the remote Cordova Android application and present the console in the same way the Smartphone app is presented on the device. The analysis showed that the Smartphone app utilizes an average 65MB of memory,

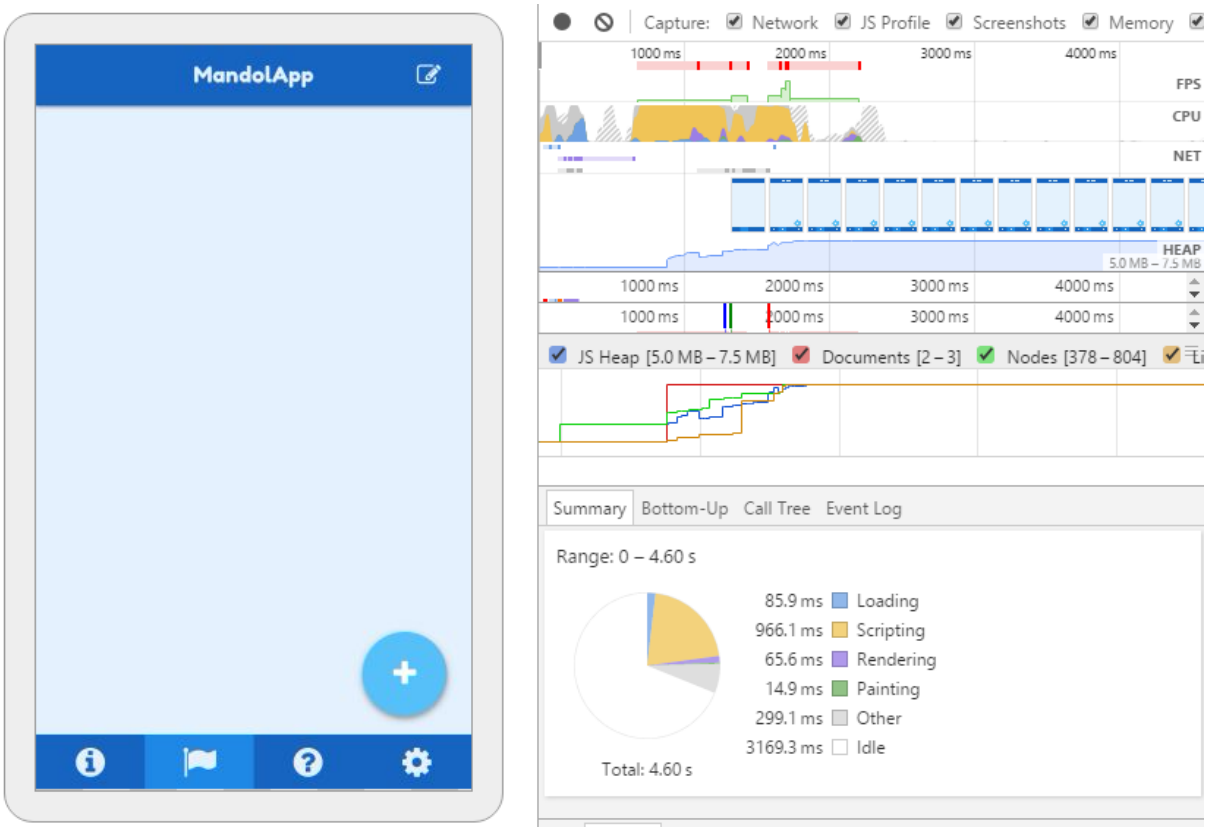

Figure 33: Chrome's Cordova application debugging interface

and starts with 93MB of internal storage usage. The necessary application's files take 84MB of internal storage space, the extra files like the cropped images used by OCR or the Tesseract language files, start with 36MB of space utilization. An analysis with no Tesseract language, with one language and two languages was done and gave the following results:

	No Language	One Language	Two Languages
Total [MB]	93.46	124	136
Application [MB]	84.22	84.22	84.22
Data [MB]	9.25	36.65	51.66

Table 4: Internal Storage analysis of Smartphone app

Based on the above results, the main factor of the Smartphone app's storage requirements is the OCR's Tesseract language data. Each user is believed to be able to speak at least two languages, so two Tesseract languages data configuration for the OCR reporting method of the Smartphone app will take a total of 136MB of the device's internal storage.

An analysis was also done on the response time of loading the four main views of the Smartphone app; the Reporting View, the Awareness View, the FAQs view and the Settings

View. On average the application takes 4.45 seconds to load, from those the 3 seconds being idle. A more detailed analysis on these numbers can be found in Table 5.

View	Time Analysis [ms]							Heap [MB]
	Loading	Scripting	Rendering	Painting	Other	Idle	Total	
Reporting View	80.4	710.1	48.4	11.5	279.7	2697.4	3830	4.4 – 6.0
Awareness View	102.3	1078	131.7	43.8	426.7	2946.5	4730	4.7 – 8.1
FAQs View	99.9	812.9	60.1	12.3	362	3269.9	4620	4.4 – 6.9
Settings View	95.6	1010.8	54.6	9.0	439.7	3099.3	4710	3.5 – 6.0

Table 5: Response time analysis on Smartphone app

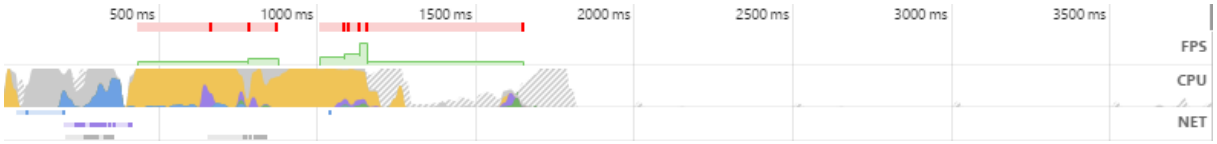

Figure 34: Reporting View response timeline

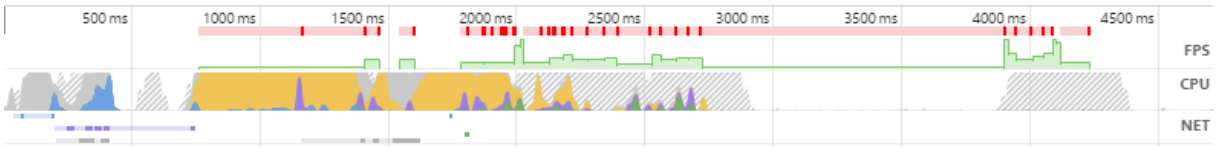

Figure 35: Awareness View response timeline

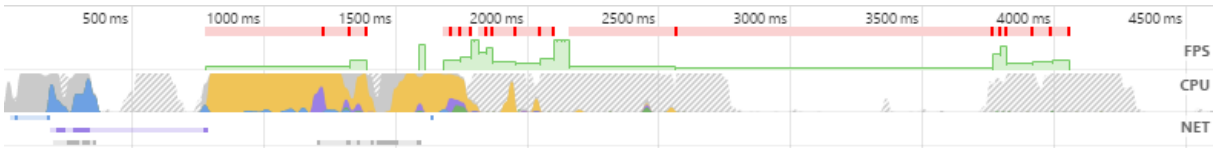

Figure 36: FAQs View response timeline

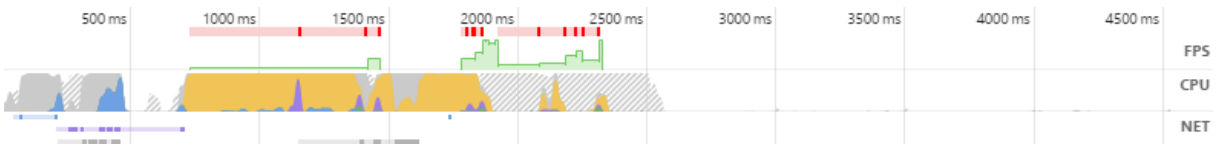

Figure 37: Settings View response timeline

8 Conclusion

In this deliverable, a detailed description of the design and development of Smartphone app is provided and the application's different views are described. The main concepts of the application are defined, being the user's anonymity, experience and awareness, and followed through the entire development process. The architecture of the Smartphone app and the user functionalities provided by the API are documented. Furthermore, an extensive analysis on the user experience and the reporting methods of the application are described and documented, including use cases with the most popular social media mobile applications. The Smartphone app's API can be integrated with third party authorities, responsible for addressing online hate speech incidents, and provide them with this information.

9 References

[1] M. M. Becky Gardiner, I. Anderson, J. Holder, D. Louter and M. Ulmanu, "The Guardian," 12 April 2016. [Online]. Available: https://www.theguardian.com/technology/2016/apr/12/the-dark-side-of-guardian-comments.

[2] A. Toor, "The Verge," 20 December 2016. [Online]. Available: http://www.theverge.com/2016/12/20/14022472/facebook-hate-speech-moderation-germany-migrant-refugee.

[3] J. Rutenberg, "NY Times," 2 Octomber 2016. [Online]. Available: https://www.nytimes.com/2016/10/03/business/media/on-twitter-hate-speech-bounded-only-by-a-character-limit.html?_r=0.

[4] N. Olivarez-Giles, "WSJ," 24 August 2016. [Online]. Available: https://www.wsj.com/articles/why-twitter-cant-shake-its-harassment-problem-1472045224.

[5] A. Cullen, Interviewee, *StatCounter - Mobile and tablet internet usage exceeds desktop for first time worldwide..* [Interview]. 1 November 2016.

[6] Wikipedia, "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Wayback_Machine.

[7] S. Rubab, B. Dhupia, B. Jaafar and N. Litayem, "Investigating User Requirements for Mobile Educational App Impact of Requirements Gathering on Software Development.," *International Journal of Engineering Research & Technology - IJERT,* vol. 4, no. 03, p. 10, 2015.

[8] MANDOLA Project, "MANDOLA Project - FAQ on Responding to on-line hate speech," July 2016. [Online]. Available: http://mandola-project.eu/m/filer_public/1a/af/1aaf50d3-8a38-40f4-b299-9c343f16cea1/mandola-d41.pdf.

[9] MANDOLA Project, "MANDOLA Project - MANDOLA Monitoring Dashboard," September 2016. [Online]. Available: http://mandola-project.eu/m/filer_public/06/b9/06b92efd-cce2-4204-a2a9-5eb2c34912f7/mandola-d31.pdf.

[10] Apache Cordova, [Online]. Available: https://cordova.apache.org/.

[11] R. W. Smith, "An Overview of the Tesseract OCR Engine," in *ICDAR*, Washington, DC, 2007.

[12] "Countering hate speech online," 28 11 2012. [Online]. Available: http://eeagrants.org/News/2012/Countering-hate-speech-online.

[13] "NGINX," NGINX Inc, 2004. [Online]. Available: https://www.nginx.com/resources/wiki/.

[14] Gunicorn, “Green Unicorn, a Python WSGI HTTP Server for UNIX,” 2010. [Online]. Available: http://gunicorn.org/.

[15] A. Ronacher, “Flask Framework, a Python Microframework,” 2010. [Online]. Available: http://flask.pocoo.org/.