Bachelor’s Thesis

DEVELOPMENT OF A SMARTPHONE APPLICATION

FOR REPORTING ONLINE HATE SPEECH

Andreas Iacovou

UNIVERSITY OF CYPRUS



COMPUTER SCIENCE DEPARTMENT

May 2017

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

DEVELOPMENT OF A SMARTPHONE APPLICATION

FOR REPORTING ONLINE HATE SPEECH

Andreas Iacovou

Under the supervision of:

Dr. George Pallis

Assistant Professor

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science

May 2017

**Acknowledgement**

I would like to express my sincere and deep gratitude towards my supervisor Dr. George Pallis for the support and guidance he offered me throughout the course of this thesis project. Dr. Pallis was always available to discuss and offer me assistance while also pushing me to achieve greater things.

Furthermore, I would also like to express my appreciation to Mr. Demetris Paschalides, who as well, provided me with valuable support throughout my thesis project.

Finally, special thanks should go to all the people that helped me with the gathering of twitter reports as well as the beta testing of the application, by using the application and reporting twitter posts related to hate speech.

**Abstract**

Living in the 21st century, we have recently seen a huge rise in the use of social media as well as the internet in general. Facebook, Instagram, Twitter and other social media websites have managed to penetrate into the life of almost a 3rd of the world’s population according to some sources (about 2.62 billion in 2018), with a projected increase in the next 3 years. [1]

Hate speech seems to be thriving under the circumstances created by such huge rise in social media usage, leading to the need of taking more drastic measures against it.

This document aims to provide the reader with details about the methodologies used for the design and implementation of a smartphone application, for reporting online hate speech found on social media. It also aims to be used as a how-to-user guide, describing the methodologies used for the extraction of requirements, the gathering of information and the testing of the new application as well as the reasons behind each choice in the design and implementation phases.

Table of Contents

[Chapter 1 **Introduction** 1](#_Toc515232925)

[1.1 Motivation 1](#_Toc515232926)

[1.2 Contribution 2](#_Toc515232927)

[1.3 Structure of this Thesis 3](#_Toc515232928)

[Chapter 2 **Literature and Related Work** 6](#_Toc515232929)

[2.1 Literature & Related Work Review 6](#_Toc515232930)

[2.1.1 The Mandola Project 6](#_Toc515232931)

[2.1.2 Related Work Comparison 9](#_Toc515232932)

[2.2 Tools 14](#_Toc515232933)

[Chapter 3 **Requirements Specification** 18](#_Toc515232934)

[3.1 Methodology 19](#_Toc515232935)

[3.2 Stakeholder Identification 20](#_Toc515232936)

[3.3 Functional Requirements 21](#_Toc515232937)

[3.4 Non-Functional Requirements 24](#_Toc515232938)

[3.5 Use Cases 25](#_Toc515232939)

[Chapter 4 **Architecture and Design** 31](#_Toc515232940)

[4.1 Architecture 31](#_Toc515232941)

[4.2 Design 35](#_Toc515232942)

[4.2.1 Design Concepts 35](#_Toc515232943)

[4.2.2 Prototype 35](#_Toc515232944)

[Chapter 5 **Development** 44](#_Toc515232945)

[5.1 Smartphone Application 44](#_Toc515232946)

[5.1.1 Database 44](#_Toc515232947)

[5.1.2 Modules and Interfaces 46](#_Toc515232948)

[5.1.3 Compatibility 60](#_Toc515232949)

[5.2 REST API 61](#_Toc515232950)

[5.2.1 Database 62](#_Toc515232951)

[5.2.2 API – application communication: 63](#_Toc515232952)

[Chapter 6 **Evaluation** 64](#_Toc515232953)

[6.1 Performance Metrics 64](#_Toc515232954)

[6.2 Beta Testing 69](#_Toc515232955)

[6.2.1 Questionnaire 70](#_Toc515232956)

[Chapter 7 **Statistical Analysis** 78](#_Toc515232957)

[7.1 Methodology 79](#_Toc515232958)

[7.2 Statistics 80](#_Toc515232959)

[Chapter 8 **Conclusion** 85](#_Toc515232960)

[8.1 Conclusion 85](#_Toc515232961)

[8.2 Future Work 86](#_Toc515232962)

[**Bibliography** 87](#_Toc515232963)

[**Appendix** A-1](#_Toc515232964)

[A. Appendix A A-1](#_Toc515232965)

[B. Appendix B B-6](#_Toc515232966)

Table of Figures

[Figure 1. Previous application architectural diagram [5] 8](#_Toc515233428)

[Figure 2. Grouping of Related Work based on characteristics 9](#_Toc515233429)

[Figure 3. Related Work Characteristics Comparison Table 10](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233430)

[Figure 4. Make Report Use Case 25](#_Toc515233431)

[Figure 5. Make Twitter Report Use Case 26](#_Toc515233432)

[Figure 6. Make Facebook Report Use Case 26](#_Toc515233433)

[Figure 7. Make Browser Report Use Case 27](#_Toc515233434)

[Figure 8. Make Picture Report Use Case 27](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233435)

[Figure 9. Save Report Use Case 28](#_Toc515233436)

[Figure 10. Filter Saved Reports and Report History Use Case 28](#_Toc515233437)

[Figure 11. Send Saved Report Use Case 29](#_Toc515233438)

[Figure 12. Delete Saved Reports and Report History Use Case 29](#_Toc515233439)

[Figure 13. View Sent Reports Statistics Use Case 30](#_Toc515233440)

[Figure 14. Application Architecture 34](#_Toc515233441)

[Figure 15. Report Main View Mock-up 36](#_Toc515233442)

[Figure 16. Report View Picture Mock-up 37](#_Toc515233443)

[Figure 17. Report View Online Report Instructions Mock-up 38](#_Toc515233444)

[Figure 18. Report View Online Report Mock-up 39](#_Toc515233445)

[Figure 19. Statistics View Mock-up 39](#_Toc515233446)

[Figure 20. Saved Reports Mock-ups 40](#_Toc515233447)

[Figure 21. Report History Mock-ups 41](#_Toc515233448)

[Figure 22. About Mock-up 41](#_Toc515233449)

[Figure 23. Settings Mock-up 42](#_Toc515233450)

[Figure 24. Help Mock-ups 43](#_Toc515233451)

[Figure 25. Database: Saved Reports Table 45](#_Toc515233452)

[Figure 26. Database: Sent Reports Table 45](#_Toc515233453)

[Figure 27. Main Module Interface 46](#_Toc515233454)

[Figure 28. Background Service-Mini App Module Implementation 48](#_Toc515233455)

[Figure 29. Notification Module Implementation 48](#_Toc515233456)

[Figure 30. Report Module Implementation 51](#_Toc515233457)

[Figure 31. Reporting Module Interface 51](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233458)

[Figure 32. Saved Reports Interface 53](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233459)

[Figure 33. Report History Module Interface 54](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233460)

[Figure 34. About Module Interface 55](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233461)

[Figure 35. Statistics Module Interface 56](#_Toc515233462)

[Figure 36. Settings Module Interface 57](#_Toc515233463)

[Figure 37. Introduction Module Interface 58](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233464)

[Figure 38. Help Module Interface 60](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233465)

[Figure 39. Smartphone Models & Android versions with no Usage Access Permission [19] 61](#_Toc515233466)

[Figure 40. TwitterReports, FacebookReports & BrowserReports REST API tables 62](#_Toc515233467)

[Figure 41. Picture Reports REST API table 62](#_Toc515233468)

[Figure 42. Performance Metrics Device Specifications 65](#_Toc515233469)

[Figure 43. CPU Usage While Loading Content 66](file:///C:\Users\andri\Documents\Uni\8o%20eksamino\Dissertation\final\AndreasIacovou_dissertation_final.docx#_Toc515233470)

[Figure 44. Memory Usage While Loading Content 67](#_Toc515233471)

[Figure 45. Time to Load Each Kind of Content 67](#_Toc515233472)

[Figure 46. Time to Send Each Kind of Report 68](#_Toc515233473)

[Figure 47. Background Service CPU Usage 69](#_Toc515233474)

[Figure 48. Background Service Memory Usage 69](#_Toc515233475)

[Figure 49. Questionnaire Question 1 Answers 71](#_Toc515233476)

[Figure 50. Questionnaire Question 2 Answers 71](#_Toc515233477)

[Figure 51. Questionnaire Question 3 Answers 72](#_Toc515233478)

[Figure 52. Questionnaire Question 4 Answer 73](#_Toc515233479)

[Figure 53. Questionnaire Question 5 Answer 73](#_Toc515233480)

[Figure 54. Questionnaire Question 6 Answers 74](#_Toc515233481)

[Figure 55. Questionnaire Question 7 Answer 75](#_Toc515233482)

[Figure 56. Questionnaire Question 8 Answer 75](#_Toc515233483)

[Figure 57. Questionnaire Question 9 Answers 76](#_Toc515233484)

[Figure 58. Questionnaire Question 10 Answers 76](#_Toc515233485)

[Figure 59. Questionnaire Question 11 Answers 77](#_Toc515233486)

[Figure 60. Questionnaire Question 12 Answers 77](#_Toc515233487)

[Figure 61. Questionnaire Question 13 Answers 78](#_Toc515233488)

[Figure 62. Reports per Hate Speech Percentage Bar Chart 81](#_Toc515233489)

[Figure 63. Reports per Category Bar Chart 81](#_Toc515233490)

[Figure 64. Reports per Country Bar Chart 82](#_Toc515233491)

[Figure 65. Reports per Time Period Bar Chart 83](#_Toc515233492)

[Figure 66. Reports per Category & Time Period Heatmap 84](#_Toc515233493)

# Chapter 1

**Introduction**

* 1. Motivation
  2. Contribution
  3. Structure of this Thesis

­­­­­­­

## Motivation

As of late, we have observed a colossal ascent in the utilization of online social networking which has laid a path for the rise of hate speech as well. Online social networking is overflowing with hate speech use. A brief look through the comments section of a political twitter post, for example, will quickly reveal the pervasiveness and depth of the issue.

A useful definition on what is hate speech can be found in the oxford dictionary. The oxford dictionary describes hate speech as an “Abusive or threatening speech or writing that expresses prejudice against a particular group, especially on the basis of race, religion, or sexual orientation.” [2]. A more challenging task however would be to identify what kind of hate speech is considered as being illegal.

The legality and nature of hate speech is complicated. Freedom of speech is a basic human right in most democratic countries around the world and the EU. Therefore, different legislations among different countries of the world have different approaches on what could be considered as freedom of speech and the restrictions and limits imposed to it that are enforced by law.

Although there is no universally acceptable legal definition of hate speech, we could consider the views of the Committee of Ministers of the Council of Europe, which states that “hate speech shall be understood as covering all forms of expression which spread, incite, promote or justify racial hatred, xenophobia, anti-Semitism or other forms of hatred based on intolerance, including: intolerance expressed by aggressive nationalism and ethnocentrism, discrimination and hostility against minorities, migrants and people of immigrant origin”. [3] [4]

Major social media companies such as Facebook and Twitter have their own definition of hate speech. They also have their own approach and regulations regarding the sort of hate speech allowed on their platforms, as well as their own reporting tools. These policies however, are often inconsistently applied and can be difficult to understand, according to users.

Although a code of conduct was also signed, between Microsoft, Twitter, Facebook and YouTube with the EU, in May 2016, to review most complaints within a 24-hour timeframe, the companies managed to meet that target in only 51% of the cases, by May 2017, according to the European Commission. [5]

In an effort to identify solutions for curtailing hate speech in social media, the MANDOLA project was born. MANDOLA, that stands for Monitoring and Detecting Online Hate Speech, is a project co-funded by the Rights, Equality and Citizenship (REC) Programme of the European Commission. The project aims at improving the public understanding of how on-line hate speech prevails and spreads, while empowering citizens to report hate speech. [5] This dissertation will present the design and implementation of a smartphone application, that has been developed for the MADNOLA project, to provide users with a friendly, fast and convenient way to anonymously report hate speech content found on social media, to relevant authorities.

## Contribution

As already discussed in the previous section, although many social media companies have their own platforms and reporting tools they seem to have failed to tackle the hate speech issue properly. Many 3rd party online platforms exist, that allow the user to report hate speech content found on social media, however they too seem to not be able to tackle the issue in a satisfactory manner.

Since nearly 80% of social media time is spent on mobile devices, according to comScore’s 2016 U.S. Cross-Platform Future in Focus, users have the tendency to ignore the reporting of hate speech, as the use of 3rd party online platforms is difficult and time consuming on such devices. [6]

The application described in this thesis aims at helping in the fight against online hate speech by making the following contributions:

* Help encourage users to report hate speech content. This is expected to be done by solving the inconvenience for users, of reporting on platforms not designed for mobile devices, or proving to be time consuming. The application is specifically created for android devices and therefore smartphones, with the main concept being the anonymity of the users as well as the enhancement of the user satisfaction and experience during the reporting process. Additionally, the application provides a universal way of reporting across different social media platforms. Users who spend most of their time on social media using mobile devices, are expected to be encouraged in reporting hate speech, rather than ignoring the reporting process due to inconvenience that other platforms or tools may present.
* Help the monitoring of hate speech reports, by solving the problems presented when reporting to social media platforms. Since social media platforms seem to be unable to cope with the vast number of reports they receive, regarding hate speech content, the application can contribute by allowing the reporting of hate speech to other relevant authorities or organizations. Hence, hate speech reports will be tackled more easily and effectively as more authorities will be involved in their monitoring.
* Raise the awareness of the users regarding hate speech. The application presents information to the users regarding hate speech, such as what is hate speech, different hate speech categories and different authorities involved in its monitoring, as well as an FAQ section. This information, along with the involvement of users in the reporting of hate speech content and the presentation of statistics about each user’s reports by our application, are expected to raise the awareness of the population on the topic of hate speech and on the most suitable ways to respond to it.

## Structure of this Thesis

The structure of this thesis is as follows:

**Chapter 1:** Introduction

The first chapter contains information about the topic of this Thesis. It makes the introduction to the hate speech problem, how it correlates to the rise of social media usage, and how approaches to deal with the issue have thus far failed. A need for an alternative solution is expanded upon, and how our application is expected to make a contribution towards this solution is further explained.

**Chapter 2:** Literature and Related Work

The second chapter consists of 2 parts. First, previous work related to our application is shortly presented, describing the characteristics of each work and highlighting its negative and positive traits.

Lastly, some of the tools used for the development of the application are presented while explaining the reasons behind the choice to use them.

**Chapter 3:** Requirements Specification

In this chapter the requirements of the application are analyzed. The methodology used for the extraction of the requirements is explained. The functional as well as the non-functional requirements are presented in the form of natural language. Lastly, use case diagrams explain the functional requirements of the application diagrammatically.

**Chapter 4:** Architecture and Design

This chapter describes the architecture of the application the as well as the different modules that it consists of. The functionality and purpose of this modules is further analyzed and described. The architecture diagram of the application is also presented.

In the next section of this chapter the design principles will be discussed. The prototype of the application will be presented and the reason behind each choice of mock-up view will be explained.

**Chapter 5:** Development

This chapter is divided into two parts, describing the development of the application as well as the development of the REST API.

The first part describes the development of the application’s database and the application modules and interfaces.

The second part describes the development of the REST API database and how it communicates with the application.

**Chapter 6:** Evaluation

This chapter describes the methodologies used for the evaluation of the application. It describes the performance metrics performed on our application, as well as the whole procedure of beta testing that took place and the feedback received by the users.

**Chapter 7:** Statistical Analysis

As a part of this project, a statistical analysis was performed on real twitter reports gathered form users.

The methodology used for the extraction and calculations of these statistics as well as statistics in the form of bar charts and heatmaps are presented.

**Chapter 8:** Conclusion

In this chapter, our conclusions about the whole the methodologies and the content provided in throughout this thesis are presented and future work that could be done is discussed.

# Chapter 2

**Literature and Related Work**

2.1 Literature & Related Work Review

2.2 Tools

## Literature & Related Work Review

### The Mandola Project

As stated in chapter 1, the application is designed and developed under the scope of the Mandola project. In order to better understand the topic of hate speech as well as the needs of the application, extensive research of the Mandola deliverables was carried out.

**Mandola project background:**

As previously described, Mandola stands for Monitoring and Detecting Online Hate Speech and is a project co-funded by the Rights, Equality and Citizenship (REC) Programme of the European Commission. The project aims at improving the public understanding of how on-line hate speech prevails and spreads, while empowering citizens to report hate speech. More specifically the objectives of the project are to monitor the spread and penetration of online hate-related speech within the EU, to provide policy makers with information that could be used for the promotion of policies regarding hate speech, to provide citizens with tools to help them tackle online hate speech and to create a reporting infrastructure for the reporting of illegal hate speech. [5]

**Definition of illegal hatred and implications deliverable:**

The Mandola deliverable on the Definition of illegal hatred and implications was first examined. The aforementioned deliverable describes in detail the serious of implications presented in terms of contents to be taken into account by the Mandola project, actions that should be taken against hate speech and in terms of the ideal definition of illegal hate speech. By studying this deliverable, we were able to further understand what can be considered as hate speech and illegal hate speech, and therefore the content that the users of the application are expected to report. [7]

**Smartphone app deliverable:**

The aforementioned deliverable presents the implementation of a smartphone application, already developed for the Mandola project, to be used for reporting hate speech content. The details of the design and implementation are extensively described and explained. [5]

The application allows users to report hate speech anonymously using two different ways:

* The application runs in the background and listens to copy events. The user copies the link of the post that he wishes to report and the application captures the copied link. Afterwards the user must manually take a screenshot of the post that he wishes to report. The user will then press the application bubble that appears on the screen of the device, similarly to how Facebook Messenger chat heads appear, and he will be redirected to the main application to make the report. He will be asked to crop the screenshot taken and an OCR (Optical Character Recognition) software will try to recognize the text contained in the screenshot. The user will also be asked to provide a title for the report as well as the categories of hate speech that the report may fall under. After that, the report is ready to be sent.
* The second way is different from the first in the procedure used to extract the text of the report. The user still copies the link of the post that he wishes to report, but instead of taking a screenshot of the report, the MANDOLA proxy server is used to mine the text. More specifically, the application will launch what is called an InAppBrowser, a web browser within the application, that will load the copied link using the MANDOLA proxy server. The user will then highlight the hate speech content. The rest of the reporting procedure after the selection of the text remains the same. This method is mostly used for the reporting of the public content where the link of the post can be loaded without the need of user credentials and so an Optical Character Recognition software is not needed to extract the text.

The reports submitted through the application are stored in a local SQLite database so the user can view previous reports. Through the MANDOLA API the reports are also stored in a Report Storage Module, an encrypted relational database.

Apart from the anonymous reporting module, the application consists of the awareness module as well. The awareness module presents statistical data to the user to raise his awareness on the online hate speech issue. [5]

Figure 2 shows the application architecture:

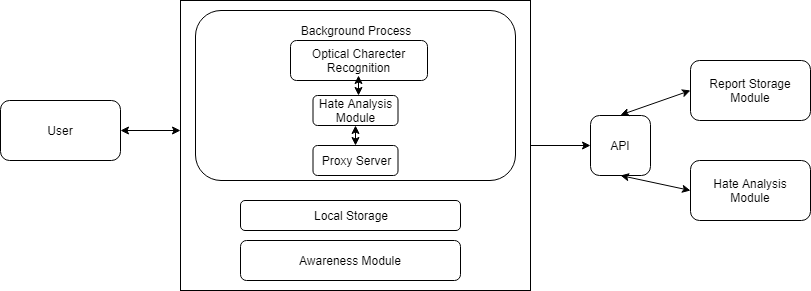


Figure 1. Previous application architectural diagram [5]

Although the application does a very fine job in allowing the user to report hate speech content, we have recognized a few downsides, as well as some new opportunities, that led to the need for a new application.

The application was created using the Cordova framework, a framework enabling the creation of hybrid smartphone applications. The application is written in HTML and JavaScript and is later compiled into android and iOS code. However, due to the complexity of the application and more specifically the bubble component that stays on top of other applications, some new Cordova plugins had to developed for android devices, while iOS devices don’t support this feature at all.

Furthermore, although the application provides a convenient way for the user to make a report, we have recognized that an even better approach to the reporting process could make it even more convenient for the user. In particular, after the user copies a link and takes a screenshot of the post he wishes to report, he is taken to the mandolapp application to finish the report. This switch between applications, the social media platform and the mandolapp, could have negative effects on the user experience. Users who may not like the idea of leaving the social media application while browsing through it may ultimately decide that switching applications to make a report is not worth it.

Additionally, some inconvenience was also spotted during the Optical Character Recognition process. A lot of times the Optical Character Recognition software seemed to have failed to recognize the text perfectly or took too long to make the recognition of the text, while all the screenshots taken when reporting had to be manually deleted by the user. [5]

### Related Work Comparison

For the creation of this application, several related applications that were developed in the past have been studied. By studying such applications, we were better able to understand the problem we are dealing with and are trying to solve. In addition, we were able to study the different approaches that each application has to offer for the solution of our problem and specify important characteristics of such applications, that should or shouldn’t be included in our application, accordingly.

For this purpose, hate speech reporting tools and applications which are specifically designed for smartphone devices were studied and their main features and characteristics were specified and analyzed.

**Grouping of Related Work based on characteristics:**

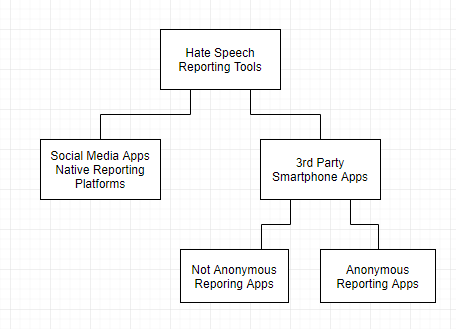


Figure 2. Grouping of Related Work based on characteristics

In the above figure, we can see the two main categories that hate speech reporting tools can be distinguished into, based on their characteristics.

Those categories are:

* **Native social media applications reporting tools**

Social media companies such as Twitter and Facebook offer their users the option to report content related to hate speech through their corresponding applications. Twitter users, for example, are able to report tweets from inside the Twitter application. However, these reporting tools are not anonymous since the reporting of content is done with the credentials of the user.

* **3rd party smartphone applications**

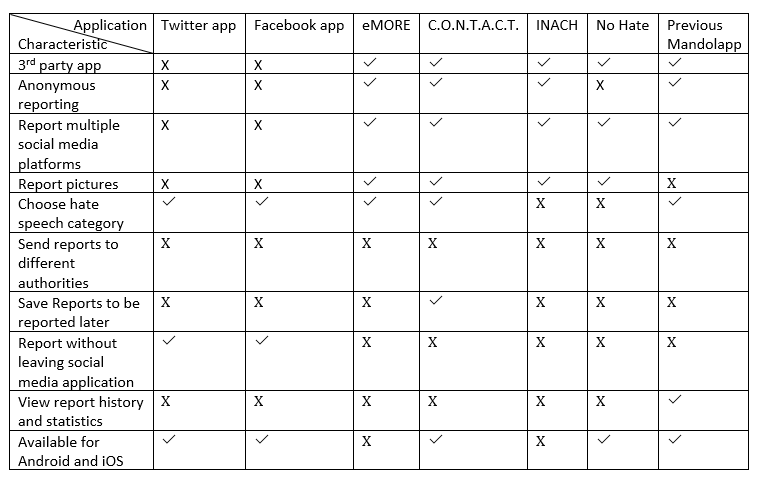
These are applications developed by 3rd party organizations, for reporting content outside the social media platforms. They are specifically developed for smartphones and they can be further distinguished into applications that offer anonymous reporting and applications that require some credentials from the user in order to make a report.

Figure . Related Work Characteristics Comparison Table

**Native social media applications reporting tools:**

* Facebook & Twitter app:

The Facebook and Twitter applications offer the ability to users to report a specific post that they may come across in the corresponding application. Users are able to specifically specify what kind of hate speech the content refers to.

As expected, the user can report without leaving the Facebook or Twitter apps, though he can only report content from that specific application and nothing else. Furthermore, the reporting is not anonymous, since the report is done while the user is logged in with his credentials.

**3rd party smartphone applications:**

Anonymous Reporting:

* eMORE:

This is a project that is also co-funded by the Rights, Equality and Citizenship (REC) Programme (2014-2020) of the EU Commission. It stands for Monitoring and Reporting Hate Speech in Europe and its aim is to contribute to developing, testing and transferring a knowledge model on online hate speech and offline hate crime. [8]

The eMORE application, which is only available on android devices, allows users to report hate speech anonymously from within the application, while also giving the user the ability to create an account and log in.

Users report by manually entering the link of the hate speech post and the social media platform that hate speech was encountered on, while also able to choose a category for hate speech, the language of the report and the country that they live in. Additionally, users can upload and report a picture.

The application doesn’t offer the ability for the user to choose where to send his reports, as all reports are sent to the same authority, neither does it offer the abilities to save reports to be reported later or view the user’s report history.

* C.O.N.T.A.C.T.:

This project is also co-funded by the European Union while its acronym stands for Creating an On-line Network, monitoring Team and phone App to Counter hate crime Tactics. The project includes partners from different EU countries, the University of Cyprus among them.

This application is available for both android and iOS devices. It gives users the ability to report, anonymously or not, hate speech encounters by providing a link, and/or a photo or other type of attachment, or a location that a hate speech encounter took place and some additional details like the description of the encounter and the category of hate speech. The application also gives the users the ability to save a report to be sent later on. [9]

Similar to other applications examined, the users must leave the social media application in order to make a report.

* INACH:

Another project co-funded by the EU, INACH stands for the International Network Against Cyber Hate. Its goal is to encourage international and local organizations and individuals on the implementation of human rights on the internet. It was founded on October 4, 2002 by Jugendschutz.net and Magenta Foundation. [10]

The INACH application is only available on the android operating system. The application offers users the ability to send hate speech reports anonymously. Users can report content from different social media platforms, by manually giving the link as well as the text of the content. Users also have the ability to report pictures and video or audio content.

The application however, doesn’t offer the ability to the user to choose the hate speech category, save a report for later, or view his report history.

Non-anonymous Reporting:

* No Hate:

The No Hate movement is a youth campaign of the Council of Europe**for human rights online**, to reduce the levels of acceptance of hate speech and develop online**youth participation and citizenship**, including in Internet governance processes. [11]

The application, which is available on both android and iOS, is in French. It does not allow for anonymous reporting, since users must give a lot of personal information in order to make a report, such as their full name, their physical and email address, their phone number etc.

Using this application, users can report Facebook, Twitter and Google+ content, as well as upload and report pictures. The application does not give the ability to the user to choose the hate speech category or to send the report to authorities other than the Hate Speech Movement. In addition, the users can’t view saved reports or statistics, while just like all previous 3rd party applications examined, users have to leave the social media application to make the report.

**Characteristics Used:**

* **3rd party application:** As already stated, the main categorization of the hate speech reporting tools was done based on whether they are native reporting tools of social media applications, or applications designed by 3rd party organizations for reporting to them.
* **Anonymous reporting:** The next major categories that 3rd party apps are divided into are the applications offering anonymous reporting and those that require some form of user credentials or other user information to make a report.
* **Report multiple social media platforms:** A characteristic of all 3rd party applications, that allow for report to be made for various social media platforms.
* **Report pictures:** A characteristic of most 3rd party applications that allow for a picture to be reported instead of a link or text.
* **Choose hate speech category:** A very important feature of some applications. It allows for the user to be able to choose what category of hate speech the content being reported belongs to. It can help authorities analyze the reports better and faster while raising the awareness of the user.
* **Send reports to different authorities:** A characteristic not found in any applications so far. It denotes that an application can send the reports to multiple authorities relevant with hate speech monitoring, instead of just the authority behind the development of the application.
* **Save reports to be reported later:** A characteristic only available in one application. Users are allowed to start making a report and save a draft of it. Then they can go back to the draft and finish and report it at any time.
* **Report without leaving social media application:** A characteristic not found in any of the 3rd party applications, that allows for the user to make a report without having to leave the social media application and launch the reporting application, thus greatly enhancing the user experience.
* **View report history and statistics:** Found only in the previous Mandolapp application, described in detail in the previous section, the user is able to view the reports he has sent and general statistics about the hate speech issue, that would help raise his awareness.
* **Available for android and iOS:** Denotes whether the application is available both on the android and iOS platforms. While the native social media reporting tools and most of the 3rd party applications are available on both platforms, 2 of the 3rd party applications are only available on android devices.

## Tools

For the implementation of our android application a number of different tools were used.

**Android Studio**

For the coding of our application Android Studio was used. Android Studio is the official integrated development environment (IDE) for the android operating system. It was created on top of the IntelliJ IDEA development environment by JetBrains, while additional libraries and tools specifically designed for android were added.

Android Studio supports features such as smart editing, advanced code refactoring and deep static code analysis. Additionally, Android Studio uses a build system that is based on Gradle, which provides flexibility when adding new libraries and automatic dependency resolution. The particular IDE also offers fast and easy Graphical User Interface (GUI) by providing drag-and-drop features and tools ready to be used. [12]

**Tooleap**

The need for a convenient and user-friendly reporting process got us searching for how the “draw over other apps” permission could be used. This permission allows an application to open windows using the TYPE\_SYSTEM\_ALERT, that is shown on top of all other applications. Simply put, the application can show a window that appears on top of every other application even though the application that started the window may not run in the foreground. Searching for tools that could be used to take advantage of this permission we discovered Tooleap. Tooleap is an Android SDK that makes use of the “draw over other apps” permission to provide a floating user interface that can be launched on top of other applications.

Tooleap functionality can be distinguished in two parts:

* The floating bubble:

Tooleap provides a floating bubble similar to Facebook Messenger chat heads. The bubble can be moved across the screen and it remains on top of other applications, even if the activity that launched it is destroyed. The bubble can also present notifications to the user. When the user presses on the bubble a side screen is launched.

* The side screen:

Once the floating bubble is clicked the side screen slides in from the left. The side screen contains an almost typical android activity, with the only difference being that activities of the side screen must extend a class from the TooleapActivities package.

These functionalities are enclosed in what is called in the Tooleap documentation as a Mini App. [13]

**Twitter Kit**

Twitter Kit for android is a software development kit (SDK) that was specifically to designed to help with the interaction of an android application with the Twitter API. Some of the features that the Twitter Kit offers are displaying tweets, logging in to twitter or tweeting from the application and more.

Twitter Kit provides a set of individual kits that can be used for different purposes.

* TwitterCore:

It enables users to log in to Twitter via Single Sign-On and make authenticated requests to the Twitter API for retrieving tweets, users and other stuff.

* TweetUI:

It enables the embedding of Tweets in an application with the help of Tweet views.

* TweetComposer:

Enables the application to compose new Tweets and share them as well as interact with the installed Twitter application.

* Twitter + MoPub:

Makes the embedding of MoPub adds in a Twitter timeline easier. [twitter kit]

For the needs of our application we have used the TwitterCore and TweetUI components of the Twitter Kit. [14]

**Proto.io**

Before the implementation of our application we decided that the best approach was to design a working prototype.

For this prototype the tool used was proto.io. Proto.io is a website that allows for the creation of complicated, interactive and operational prototypes for android as well as iOS devices. The proto.io android application offers the ability to test the prototypes created on a real android device. [15]

**MPAndroidChart**

MPAndroidChart is a very powerful android library that enables the easy creation of chart and graph views. Currently, the library supports the creation of line, bar, pie, radar, bubble and candlestick charts as well as scaling, dragging and animations. [16]

In particular, the library was used to create pie charts for the presentation of some statistics in our application.

**Spark Java**

Spark Java is a microframework that is used for creating web applications and REST APIs in java 8 with minimal effort. It was developed as an alternative for Java developers that want to develop their web applications as easy as possible. By default, it runs on an embedded Jetty web server but can be configured to run on other applications.

The main difference of Spark from other similar frameworks is that instead of following the MVC pattern used in many of them, such as Spring MVC, it revolves around the idea of micro-services and rabid application development. This is also what makes it a very lightweight framework since no unnecessary libraries, classes or code is needed, being able to launch a REST API within a few lines of code. [17]

Another advantage is that Spark is a framework created for the Java programming language. Since android applications are written in Java as well there is the obvious advantage of being able to write code in the same programming language environment.

The Spark Java framework was used to create the REST API that our application will communicate with to send the reports of the users.

# Chapter 3

**Requirements Specification**

3.1 Methodology

3.2 Stakeholder Identification

3.3 Functional Requirements

3.4 Non-Functional Requirements

3.5 Use Cases

To find the specifications and requirements of our application, some extensive research took place. As mentioned in the previous chapter, the previous application of the Mandola project was carefully analyzed and tested and its positive and negative features were specified.

To better understand those positive and negative features the previous application was also given to some users to receive their feedback. Everyday users where chosen for this task since that would be the target audience of our application. Although this process was in the form of a small discussion, rather than a well-documented procedure, we received important information.

After some discussions with my supervisor the main concern that the application would need to address was specified; the easier, more convenient and more effective reporting process. Additionally, some other needs were specified as well, like the ability for the user to save reports for later, to view previous reports and statistics as well as be able to visit a help section and a settings section to be able to customize his experience.

During this analysis we found that a different approach altogether would be the best way to go, redesigning and reimplementing the new application from the beginning, utilizing the new capabilities and features that the Twitter API and Tooleap, for example, had to offer.

Since some of these new features that would make the reporting process easier are available only for the android operating system, we decided that the best way to go would be to develop a native android application.

The need for an API to be developed was also specified. The purpose of this API would be to receive the reports made by our application and store them in a database. The API could later be used for the extraction of statistics regarding hate speech reports and it could also be used by relevant authorities that wished to receive reports from our application.

Other applications related to hate speech reporting were researched as well, more details for which can be found in chapter 2. Our conclusion was that even though there were a couple of other applications aimed at hate speech reporting, none of them would meet the requirements that we had set for the new application.

This chapter defines in detail the requirements for the project. The methodology for the extraction of the requirements as well as the functional and non-functional requirements are specified.

## Methodology

For the extraction of the requirements of our application a set of steps and methodologies was used.

1. **Related Work Analysis**

The first step towards the specification of the requirements was done by studying the previous work available, as described in the previous chapter. By studying that work we were able to better understand the needs and characteristics of such applications, as well as have a look on features that probably shouldn’t be included in our own application. At the end of this phase enough material was gathered about possible characteristics of our application.

1. **Brainstorming**

After the analysis of the related work, short brainstorming sessions took place. In these brainstorming sessions are to be included discussions with common citizens as to their own experiences on reporting hate speech and their needs. Furthermore, brainstorming sessions with my academic supervisor and Mr. Paschalides took place, where the main concern that the application would need to address was specified; the easier, more convenient and more effective reporting process of hate speech found on social media. Additionally, some other needs were specified as well, like the ability for the user to save reports for later, to view previous reports and statistics, as well as be able to visit a help section and a settings section to be able to customize his experience.

1. **Stakeholder Identification**

During the brainstorming sessions the stakeholder identification took place. The stakeholders are the groups of people that are affected by our application either directly or indirectly. The stakeholders were divided into the users of our application and other groups that might be affected by its operation.

1. **Functional Requirements**

The [functional requirements](https://en.wikipedia.org/wiki/Functional_requirement) explain what has to be done by identifying the necessary tasks, actions or activities that must be accomplished by our application. The functional requirements were specified in the form of natural language as well as use case diagrams.

1. **Non-functional Requirements**

The [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement) are requirements that specify criteria that can be used to judge the operation of our application, rather than specific features and tasks that it must be able to carry out. The non-functional requirements where specified in the form of natural language.

## Stakeholder Identification

A very important step in defining the requirements of the application is to recognize the possible stakeholder of our application and their relation to our system and their needs.

**Everyday people:**

The users of our application are expected to be everyday people, with no advance knowledge in technology or android phones or even the hate speech topic. The users are, however, expected to be able to have a basic interaction with social media sites as well as a basic interaction with android devices.

The application aims at helping everyday people raise awareness on of what online hate speech is and how to confront it, while encouraging them and giving them a convenient way of reporting hate speech content found on social media.

**Law enforcement agencies & hate speech confronting authorities:**

Our application is also expected to benefit law enforcement agencies and authorities working on confronting hate speech content. Such organizations will be better able to recognize illegal material and online hate speech content to act upon.

## Functional Requirements

**Anonymously Report Hate Speech Content:**

As previously discussed, the main goal of the application is the reporting of hate speech content. Therefore, the application must provide a way to the users to report content that may come across on social media applications, while on their android phones. The reporting process has to be anonymous for the user. That means that the user won’t have to provide any sort of credentials or other kinds of personal information to make a report using our applications and no other information about the users shall be withheld by our application. Additionally, the application must give the user the option to report contents from a variety of social media applications. The user is expected to be able to:

* **Report Tweets:**

The application must provide the user with the ability to report tweets found in the twitter application while browsing the twitter application.

* **Report Facebook Posts:**

The application must allow the user to report public Facebook posts that he may encounter while using the Facebook application. Private posts are not supported, since neither the authorities that the post will be reported to, nor our application will be able to retrieve the post, due to the fact the reporting process will be anonymous.

* **Report Content Found on Internet Browser:**

Our application must also provide the ability to the user to report any other content he may come across while browsing the internet through an internet browser. Such content may be a post from social media websites accessed from a browser, articles and comments found on websites and other. The only requirement is that the content to be reported provides a link for our application to be able to retrieve it.

* **Report Pictures:**

The reporting of pictures must also be supported. Our application must provide a way for the user to upload a picture from the picture gallery of the device and report it to the API.

Furthermore, the application is expected to be able to:

* **Get post by copied link:**

Instead of the user having to manually enter the link of a post or content that he wishes to report, the application shall ideally be able to automatically get a post when the user copies its link from within the social media application, while our application is active.

* **Preview Content:**

Since the application will automatically get a post when the user copies its link, it must also provide the user with a preview of the post that he is about to report so he will be sure that he is reporting the right post or picture.

**Give Information about the Report:**

In order to help reporting authorities to more easily review and deal with a report, the application must prompt the user to enter some information about a report. The user should be able to:

* **Choose Hate Speech Category:**

The application must provide the ability to the user to choose one or more categories that the hate speech content belongs to.

The categories that the user must be able to choose from are the following:

* Religious: Content that may present hate speech towards a person’s religion.
* Gender: Content that may present hate speech towards a gender (e.g. women).
* Sexual: Content that may present hate speech towards a person’s sexual orientation, sexual harassment or sexual content that is inappropriate.
* Class: Content that may present interclass or interschool hate speech between students or other members of the school community.
* Politics: Content that may present hate speech towards politicians or political ideas.
* Ethnicity: Content that may present hate speech towards an ethnic group (e.g. black people).
* Nationality: Content that may present hate speech towards people from a different nation.
* Other: Content that doesn’t belong in any other category
* **Choose Different Authorities to Send the Report to:**

The application must be able to send the reports to a set of different authorities rather than one particular and fixed hate speech dealing authority. It must provide the ability to the user to choose one or more relevant authorities to send the report to. The authorities list will consist of the authorities and organizations that will want to participate in our project and will be using our API.

* **Give Description**

The user must have the ability to provide a short description of the report in case it’s not clear why the report was considered as hate speech by the user or if the user wants to provide any more information about the report in general.

**Save Reports for Later**

The user must be able to save a report, to be later sent, in a database. This is necessary due to the fact that the user must choose one or more categories that the hate speech in the report belongs to as well as the authorities he wishes to send it to. Additionally, many users would like to review their report to make sure that it contains hate speech before it is sent. As this is time consuming it will have a negative impact on some users reporting experience, especially while they are browsing on social media. Therefore, the user must have the ability to save a report to be sent later on, at a moment of his choosing, without the need to choose categories or authorities. This option must be given for Twitter, Facebook and internet browser reports. It was decided that the save option isn’t needed for picture reports, since the picture to be reported will be chosen from the library of the device. Hence, such a report can be made at any given time and it will have no impact on users’ social media browsing experience.

The users must also have the ability to perform a set of actions on those saved reports such as:

* **Send Saved Report:**

The user must be able to edit any information already given for the report at the time of saving it and report it at any time.

* **Filter Saved Reports:**

The user must be able to filter saved reports by their date or their description, and/or by the application they were made from. That way the user will be able to easily navigate even through very large number of reports.

* **Delete Saved Reports:**

The user must be able to delete saved reports either individually or all together.

**Save Sent Reports**

The application must save the sent reports in a database so the user must be able to view his report history. Furthermore, the users must have the ability to perform a set of actions on those reports such as:

* **Filter Sent Reports:**

The user must be able to filter sent reports by their date or their description, and/or by the application they were made from. That way the user will be able to easily navigate even through very large number of reports.

* **Delete Sent Reports:**

The user must be able to delete sent reports either individually or all together.

* **Sent Reports Statistics**

The user must be able to view certain statistics about the reports that he has sent that will help raise his awareness on the hate speech issue.

**Dismiss Bubble & Relaunch**

In order to provide the best possible user experience, the application must give the user the ability to dismiss the bubble but relaunch it at any time without the need to navigate to the main application again.

## Non-Functional Requirements

Below, the non-functional requirements of our application are going to be analyzed.

**Backwards Compatibility**

The application and by extension the libraries used should be usable on all android versions down to at least API Level 19 or KitKat. This way our application will be usable on approximately 95% of the all the android devices that have access to the Google Play Store. [18]

**Extensibility**

The application should provide easy extensibility to new developers. The categories of the hate speech content as well as the authorities that the user can report to can surely change in the future and will need to edited. Our application must make this change as easy as possible.

**Code documentation**

All public classes must provide documentation in the form of Javadoc comments. This way students, for example, who may be asked to extend the system in later years will find extending the system a lot easier.

**Lightweight Application**

Our application should be as lightweight as possible. The user is expected to use the application in parallel with other applications, while it is aimed at being always open for the user’s convenience. Although the hardware specifications such as memory and processor power of modern devices is continuously increasing, the main problem continues to be the battery capacity. This means that our application should use as few hardware resources as possible to make the user experience smoother.

## Use Cases

Below, a set of use cases will be presented that are used to diagrammatically describe the functional requirements of our application.

**Anonymously Report Hate Speech Content:**

**Make a Report:**

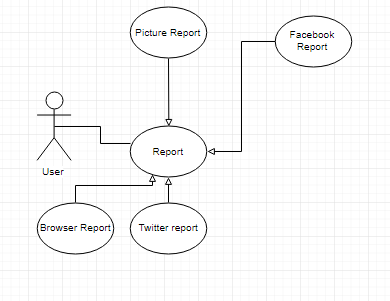


Figure 4. Make Report Use Case

The application should provide the user with the ability to report twitter, Facebook and internet browser contents as well as pictures from the library.

**Make Twitter Report:**

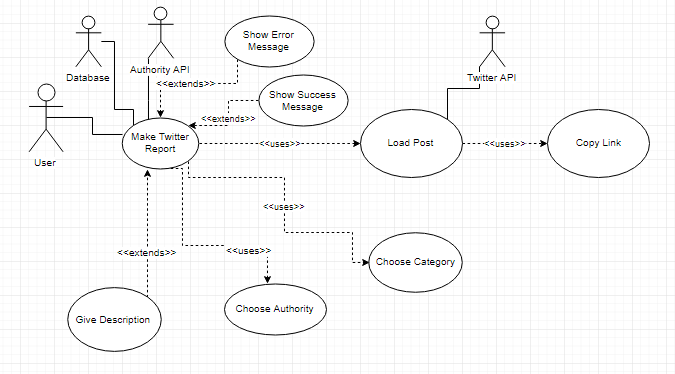


Figure 5. Make Twitter Report Use Case

**Make Facebook Report:**

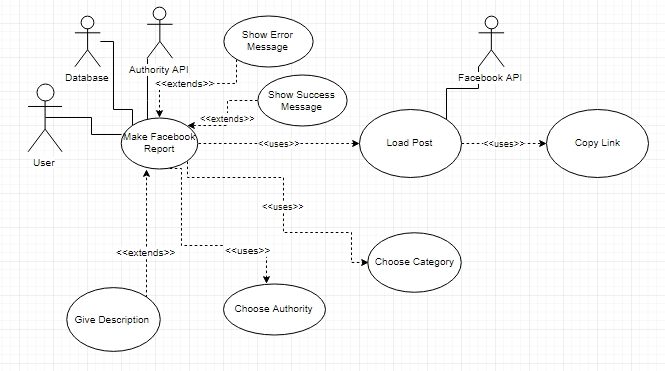


Figure 6. Make Facebook Report Use Case

**Make Internet Browser Report:**

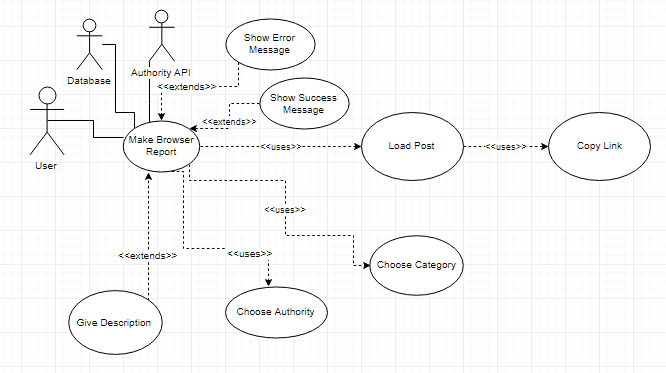


Figure 7. Make Browser Report Use Case

The user must be able to report by copying the link of a post that contains hate speech. The application must be able to load the post so the user can check the post again before reporting.

Furthermore, the application must provide the ability to the user to choose one or more categories for the report, as well as one or more authorities to send the report to. The ability to optionally add a description to the report must also be given.

The application must inform the user with a message if the report was sent successfully or not. If the sending of the report is successful then the sent report must be saved in a database.

**Make Picture Report:**

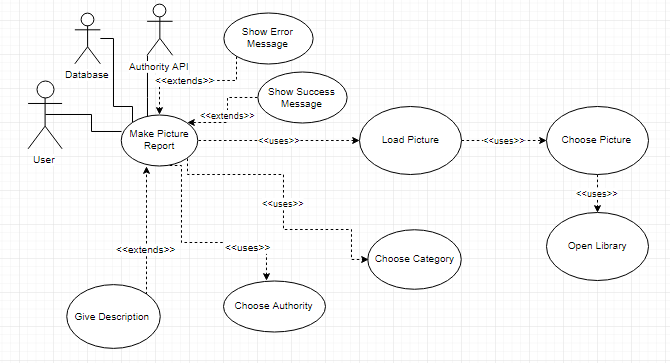


Figure . Make Picture Report Use Case

The application must also allow the user to choose a picture from the library to report. The application must load the picture for the user to preview, and once again must provide the ability to the user to choose one or more categories for the report, one or more authorities to send the report to and an optional description.

**Save Reports for Later:**

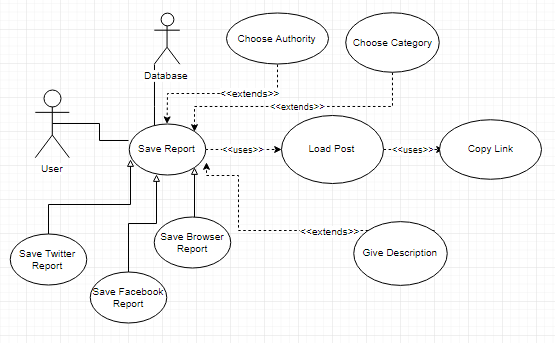


Figure 9. Save Report Use Case

As previously mentioned, the application must allow the user to save a report to be sent later in time. This is particularly useful in situations where the user doesn’t want his social media browsing experience to suffer because he has to write a description as well as choose categories and authorities for the report. Picture reports don’t provide this option since the reporting of pictures from the library can be done at any time and therefore the user experience will not suffer.

**Filter Saved Reports and Report History:**

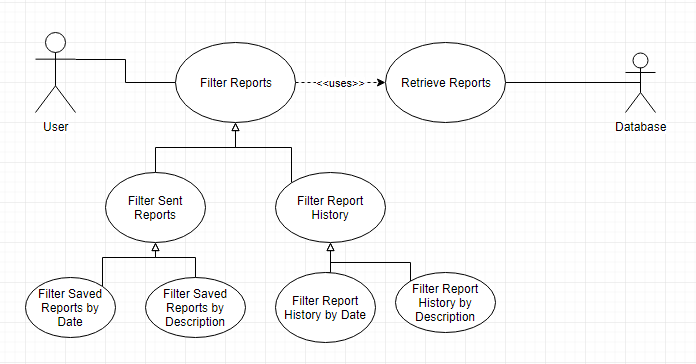


Figure 10. Filter Saved Reports and Report History Use Case

The user must be able to filter saved reports as well as the report history by date and by description.

**Send Saved Report:**

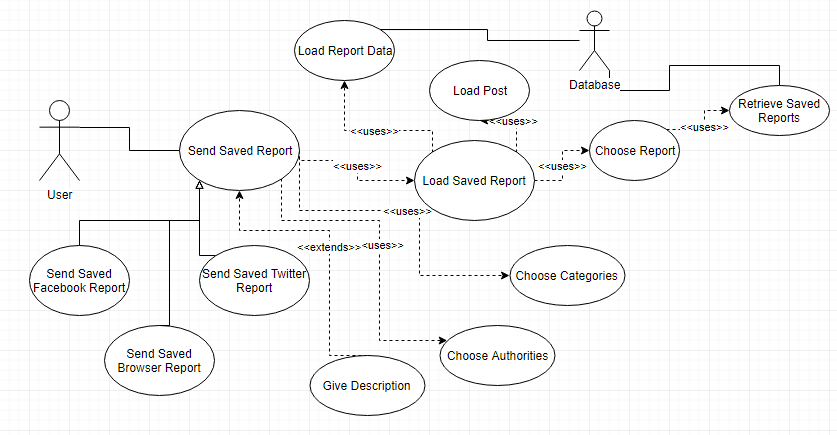


Figure 11. Send Saved Report Use Case

The user must be able to view his saved reports and choose one to send. The application must retrieve any categories, authorities and descriptions the user has given before saving and give him the ability to further change them before sending the report. The application must also load the post again for the user be able to review the post.

**Delete Saved Reports and Report History:**

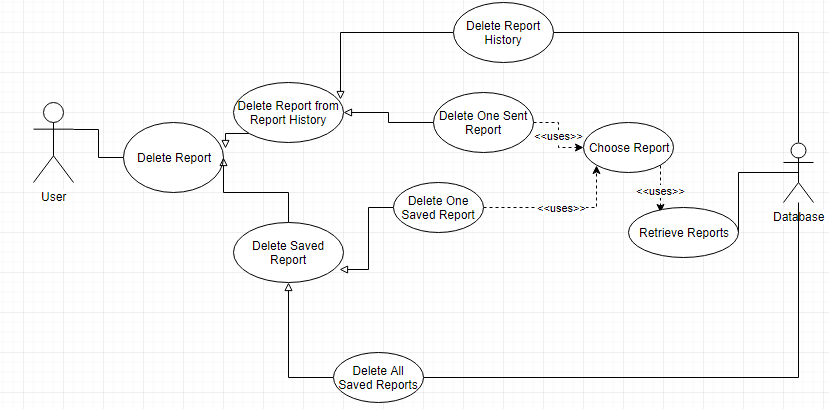


Figure 12. Delete Saved Reports and Report History Use Case

The user must be able to delete saved reports as well as reports from the report history either individually, or all together.

**View Sent Reports Statistics:**

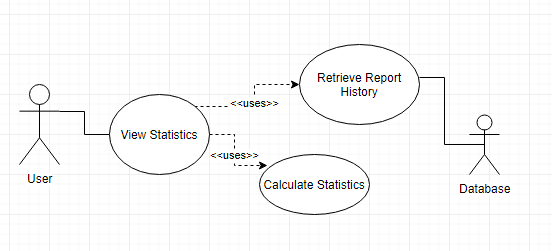


Figure 13. View Sent Reports Statistics Use Case

The application must provide certain statistics about sent reports from the report history.

# Chapter 4

**Architecture and Design**

4.1 Architecture

4.2 Design

## Architecture

The architectural design defines all the building blocks of our application, as the describe its internal design as well as the way each component communicates with each other.

Our application will consist of 8 basic modules:

* The main module
* The report module
* The statistics module
* The saved reports module
* The report history module
* The help module
* The about module
* The settings module

**Main Module:**

The main module is the entry point of our application. It is basically the first screen that the user sees when he launches the application. From the main module the user will be able to launch the mini app (bubble) created using Tooleap. It can also be used as a control point for the bubble, providing the user the ability to remove it and relaunch it. The main module is not used for anything else since all other functionality of our application is implemented in other modules.

**Report Module:**

The report module encloses the functionality used for the reporting process. It is the entry point of the mini app (bubble), meaning that it is the first screen that the user sees once he opens the mini app.

The reporting module can essentially be divided into two types. The first type is the reporting of pictures from the picture library of the device and the second type is the reporting of tweets, public Facebook posts and internet browser content.

The reporting of the picture report is a very simple process. A button is provided to the user in the first screen of the report module. When the button is pressed the bubble the side screen of the mini app (screen that appears when bubble is pressed) hides and a new android activity is launched that takes the user to the library of the device. The user can then press on any picture he likes. Once the user presses a picture the side screen will open again, loading the chosen picture and giving the user the options to choose categories, authorities and give a description, before sending the report.

The reporting of the online content is a bit more complicated process. When the bubble is launched for the first time, after the user presses the respective button from the main module, an android service is launched. The service runs continuously in the background, listening which application is open at every moment. The user is asked to open the application that he wishes to report from (e.g. Twitter). The service understands that the application in the foreground is Twitter and so a new screen is loaded in the reporting module providing the user with further instructions. The user will then copy the link of a post that he wishes to report. The android service running in the background will listen to the copy event and take that link. Afterwards, the reporting module of the mini app will take that link and load a new screen, for the user to report. In case of a Twitter or Facebook post the application will request to retrieve further information about the post from the respective API (Twitter or Facebook API) to embed it in screen, for the user to be able to preview the post. This is not needed in an internet browser report as the content can be loaded automatically from the copied link. The user will then proceed to choose categories, description and to optionally give a description before reporting the post. Otherwise, the user can save the report to be reported later without having to choose a category or an authority. In both situations the report will be saved in the corresponding table (report history or saved reports) in a relational SQLite database, which comprises the report storage module.

**Statistics Module:**

The statistics module encloses the functionality and views of the statistical analysis of the user’s report history. In order to increase the awareness of the user as well as provide him with useful information about the reports he sends the application will present to the user important statistics about his report history.

**Saved Reports Module:**

The application provides the user with a way of viewing saved reports and sending them to the authorities he chooses. The application provides a way for the user to filter saved reports either based on their date that were saved or their description. The user can also filter the reports based on the application they were from (Twitter, Facebook, browser, picture). When viewing the saved reports, the user can choose one from the list. A new screen is loaded giving the user the ability to change any category or authority selection and edit any description he might have given before saving the report in the report module.

**Report History Module:**

The application also provides the user with a way of viewing his report history. The application provides a way for the user to filter the report history either based on their date that were saved or their description. The user can also filter the reports based on the application they were from (Twitter, Facebook, browser, picture).

**Help Module:**

The help module provides information and details that are aimed at helping the user use the application correctly and easily. It encloses the FAQ (Frequently Asked Questions) section as well as helping guidelines as to what each category of hate speech means, who are the authorities the user can report to and how to make a report for each application (Twitter, Facebook, browser, picture). The help module also contains the application introduction that was later decided to be included in the application to make a short introduction to the user on how to use the application and what the application is.

**About Module:**

The about module provides a different type of information to the users. It gives information about the application and its purpose as well as the project of Mandola and the topic of hate speech in general.

**Settings Module:**

The application provides the user with the ability to customize their experience up to a specific point. From the settings module the users can choose to delete all saved reports or all of their report history. Moreover, users can choose a default authority or authorities that they wish to be preselected every time they make a report. Users can also choose to remove the reporting guidelines from the application, referring to the moving pictures (gifs) appear in our application when users open a social media application, showing them how to copy the link of a post.

**REST API:**

For the purpose of this project, a REST API was also created. The REST API is used to receive the reports of the users from our application and store them in a MySQL database or the Report Storage Module. Once the API receives a report, it stores it in the appropriate table in the Report Storage Module, according to the application that the report concerns (Twitter, Facebook, browser, picture) and return the appropriate Http message on whether the operation was successful or not.

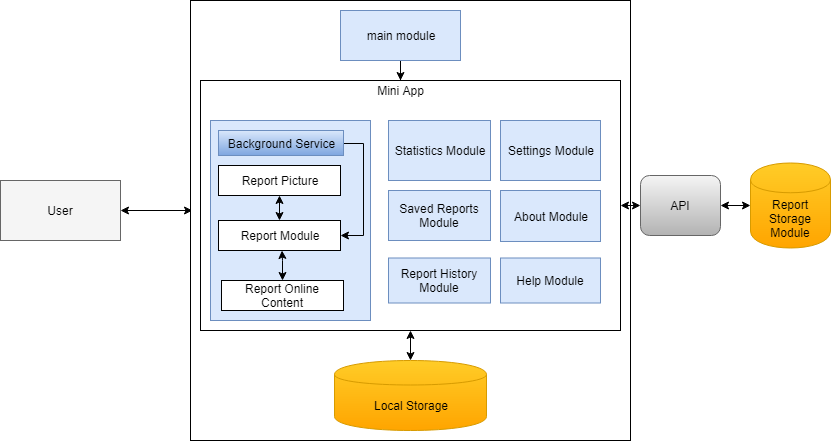


Figure 14. Application Architecture

## Design

### Design Concepts

In the Requirements Specification section, we have already specified some requirements that our application needs to meet. The designing and prototyping of our application was carefully done in order for our application to be able to meet those requirements at a later stage.

**User Experience:**

The main concern of our application, as previously discussed, is the user experience. The application must easy to use, providing a convenient way for the users to report hate speech content found on social media. This process has to be as easy and fast as possible so users will not lose interest in reporting something due to inconvenience or complicated instructions.

A key concept for making the user experience better is for the user to not have to leave the application he is currently using, switching over to our application to make the report. The best possible solution was found to be the use of Tooleap, more details for which can be found in the Literature and Related Work chapter.

With the use of Tooleap, a mini app was created enclosing most of the functionality of the application. The mini app appears to user in the form of a bubble (e.g. Facebook Messenger chat heads) that when clicked upon, opens a side screen that takes about 2/3 of the screen of the device and contains the views activities of our application like any other application screen would.

Using the bubble, the users don’t have to switch application to make a report, since both the bubble and the side screen can float on top of any other application.

### Prototype

In order to establish a prototype of the application, to get the idea of how it is going to look and behave, we used the proto.io website. Mock-up screens where created for every module of our application as well as the interactions and transitions between them, in what became a functional prototype of the application.

**Reporting Mock-ups**

The reporting module of the application consists of several views. The first view of the report module is also the first view of the mini app. It prompts the user to open a social media application or an internet browser. This view will also provide a button for the user to press in case he wants to report by uploading a picture from his library.

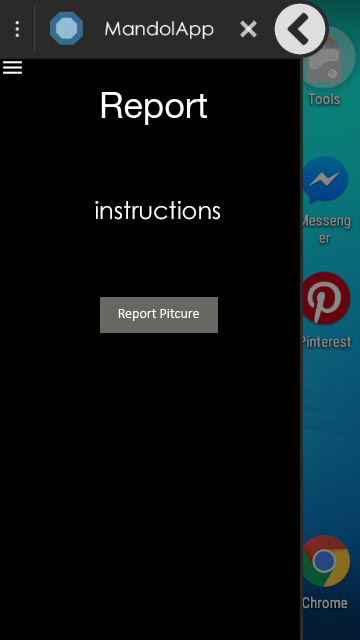


Figure 15. Report Main View Mock-up

If the user pressed the Report Picture button then the side screen will close and provide the user with the ability to choose a picture to upload. Once the user chooses the picture the side screen must open again revealing another view. The new view should present the chosen picture for the user to preview as well as the details the user must complete in order to complete the report.

Those details are the description of the report, the categories of hate speech found and the authorities that the report must be sent to. The description field is a text area that accepts multiline text so that the user won’t have to worry about the size of his description. The category and authority fields however are going to be implemented as drop down menus, or spinners as called in android, presenting to the user his options as well as a check box next to each option for the user to be able to check it. These two fields will accept more than one value meaning that the user will be able to tick many categories and many authorities. The description field was decided to be optional, whereas the category and the authority fields are obligatory, and they need to be completed for the report to be sent. The user can send the report by pressing the Report button or cancel and return to the previous screen by pressing the Cancel button or the android back button.

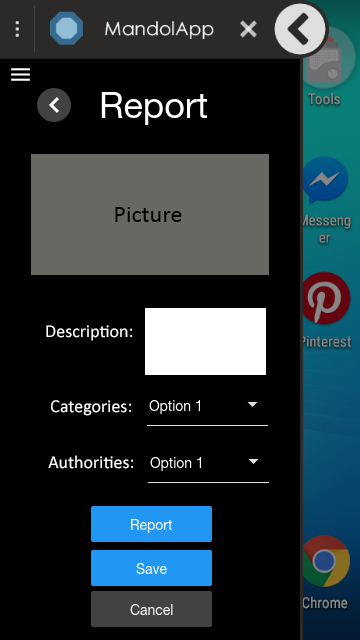


Figure 16. Report View Picture Mock-up

If the user wants to make a report using a social media post then he must open the appropriate application, for example Twitter, Facebook or an internet browser application. Our application will notice that the social media application is in the foreground and another view will be loaded providing further information to the user about how to copy a link of a post in the specific application.

The view will consist of a text giving instructions to the user on what to do next, for example copy a link, and a moving image (gif) showing the user how to copy the link. A gif will not be shown if a user uses an internet browser, since in that situation our application offers the freedom to the user to copy any link of any content that he may wish to report and so specific guidelines on how to copy a link cannot be given.

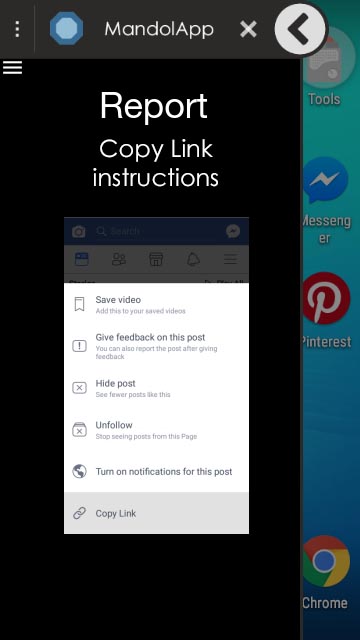


Figure 17. Report View Online Report Instructions Mock-up

When the user copies a link, the application will load another view. This view will have to load the content that the user copied so he will be able to preview it before sending the report. Furthermore, just like the Report View Picture Mock-up, the view will provide the description, category and authority fields for the user to complete before sending the report. The report is sent by pressing the Report button provided, or cancelled if the cancel button is pressed, consequently redirecting to the previous screen. The reporting procedure can also be cancelled by pressing the android back button, since many android users are familiar with this functionality.

In this scenario, unlike the picture report, the user can also choose to save the report for later by pressing the Save button, without having to complete any report details.

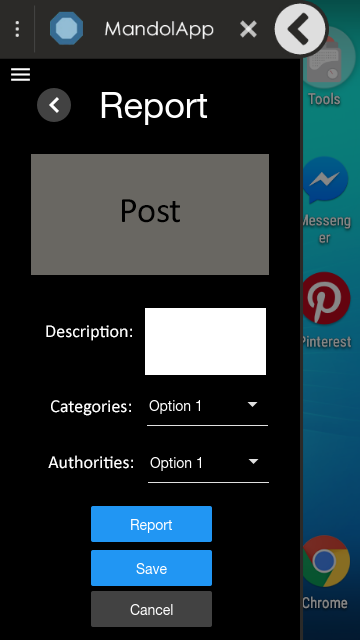


Figure 18. Report View Online Report Mock-up

**Statistics Mock-ups**

Unlike the reporting module, the statistics module consists of only a single view. When the user navigates to the statistics module then the view will be presented to him. The statistics view will contain a list of pie charts, one for each statistic (e.g. number of reports per category), along with the chart titles and description labels. Scrolling through the list the user will be able to see the different statistics.

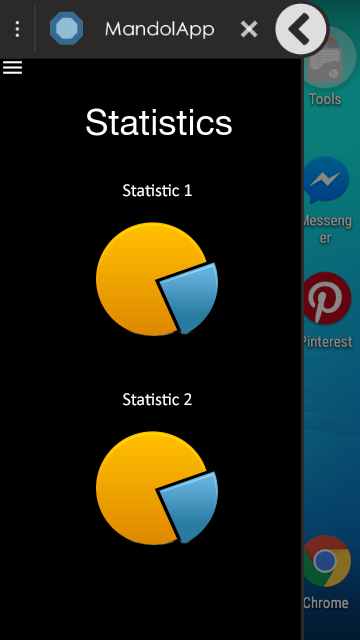


Figure 19. Statistics View Mock-up

**Saved Reports Mock-ups**

The application provides a way for the user to view his saved reports. The saved reports screen presents a list to the user containing the saved reports in a descending order starting from the most recent one. Each item on the list will consist of an icon of the application that the report is for as well as the date that the report was saved and its description, if any. At the top of the page there is also a search bar, giving the user the ability to filter and search through reports.

By pressing on a list item, the application must load another screen similar to the Report View Online Mock-up, presenting the post as well as any details of the report the user has already given and giving him the ability to further edit them. The user will be able to send the report by pressing the Report button or delete the saved report and redirected to the previous screen by pressing the Delete button.

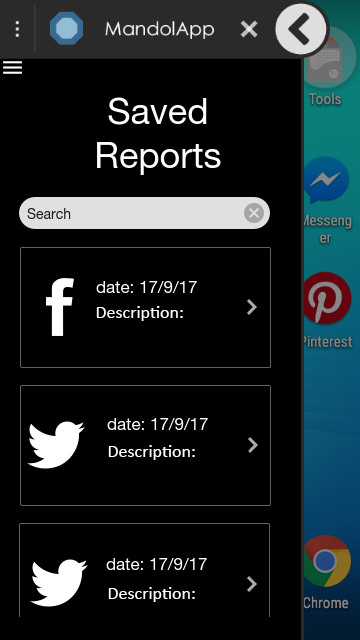
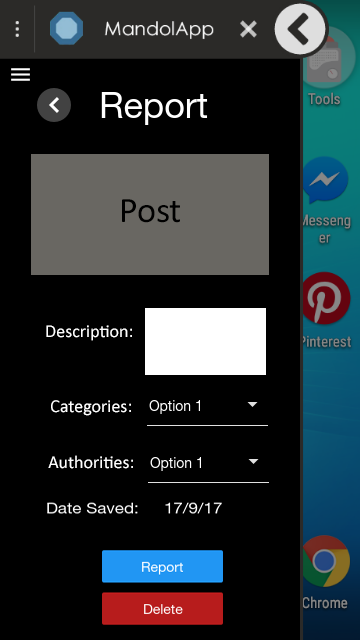
 

Figure 20. Saved Reports Mock-ups

**Report History Mock-ups**

Similar to the saved reports first screen mock-up, the application will give the user the ability to view his report history by presenting to him a list containing the reports that were sent. As in the previous mock-ups the list will contain an icon of the application the report was for and its date and description. It will also give the user the ability to filter and search from the top of the page. The user will be directed to another view if an item of the list is pressed.

The next view will load the post or picture of the report, the date that the report was sent, as well as the details of the report (description, categories, authorities) that the user gave while reporting. The user will be able to delete a report from the report history by pressing on the Clear button.

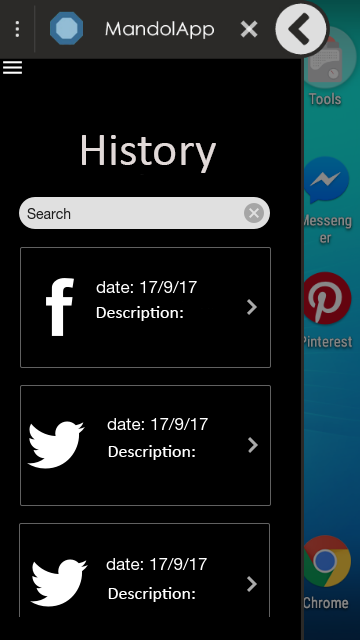
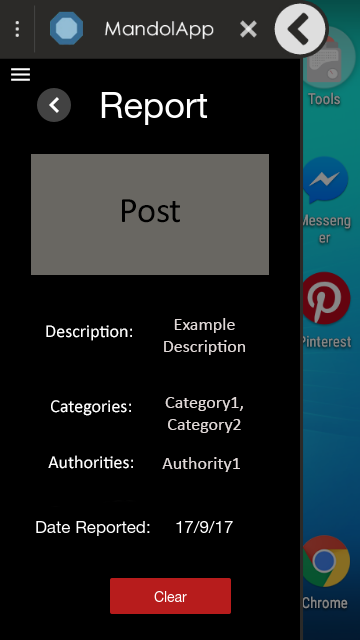
 

Figure 21. Report History Mock-ups

**About Mock-ups**

The about module will consist of a single screen the will give the user details about the Mandola project, the creation of the application and the issue of hate speech. It will also contain a link to the Mandola website for further reading.

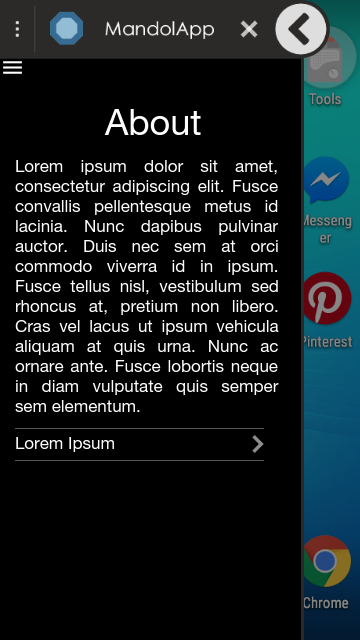


Figure 22. About Mock-up

**Settings** **Mock-up**

The application will provide screen to the user that will enable to him to customize some of the behavior of the application as well as delete the saved reports and the report history.

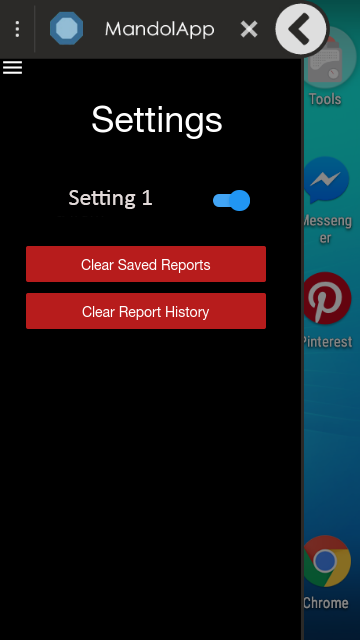


Figure 23. Settings Mock-up

**Help** **Mock-ups**

The help module is aimed at providing useful information to the user that will help him better understand how to properly use the application. The first screen of the help module will provide two options to the user. To view FAQs (Frequently Asked Questions) and to view report instruction specifically for each application.

By pressing on the FAQ button, the user will be presented with a list containing questions and their answers.

If the user presses on the report instructions button a list of buttons will appear, one for each kind of application the user can report from. By pressing on a button, the user will be redirected to another screen, containing step by step instructions on how to report using the specific application.

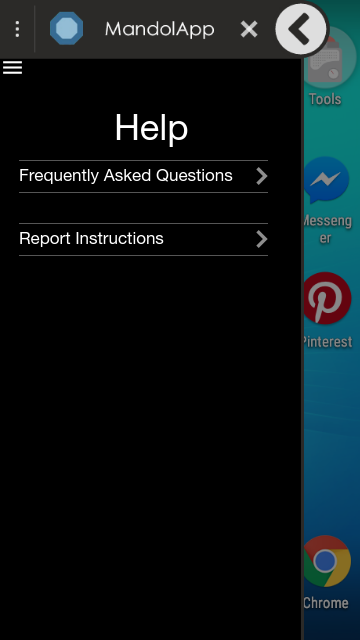
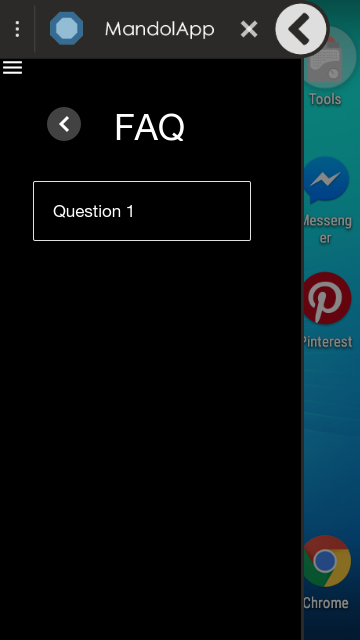
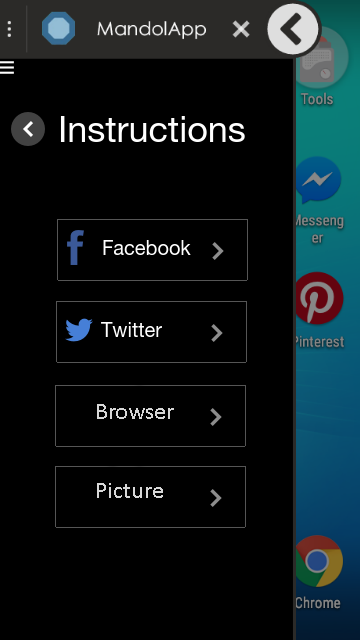
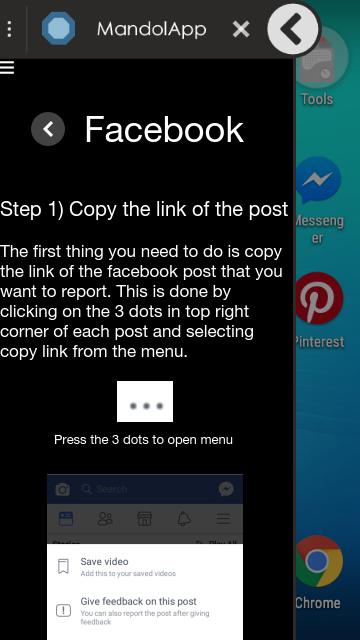
   

Figure 24. Help Mock-ups

# Chapter 5

**Development**

5.1 Smartphone Application

5.2 REST API

The development phase describes the practices, technology and tools used to develop the application, as well as the REST API. For the development of the application all specified requirements and design of the application were taken into account and implemented to the most feasible way.

## Smartphone Application

### Database

The android operating system has its own relational database that is available to developers, SQLite. The database was used for the storing of the reports of our application.

In particular, there are two kinds of reports that we would like to store in our application:

* Saved Reports: are the reports that were saved to be reported later
* Sent Reports: are the reports that were sent

Therefore, we have created two tables in the SQLite database, one for each kind of report.

**Saved Reports:**

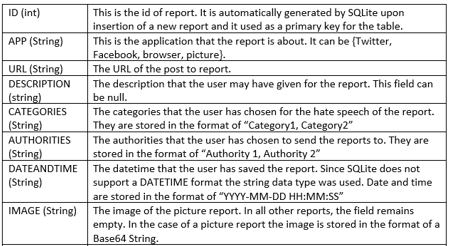


Figure 25. Database: Saved Reports Table

For extensibility reasons, it was decided that the different reports for each application (Twitter, Facebook, browser, picture) would not be stored in their own table. That is because in the future, more application may be supported for reporting and therefore new tables would have to be added as well, making the number of our tables increasingly large. The field APP contains the information about the application the report concerns.

**Sent Reports:**

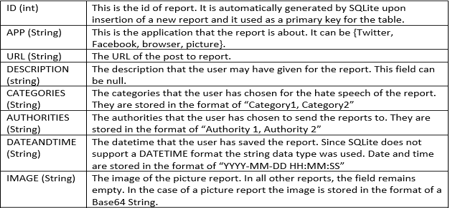


Figure 26. Database: Sent Reports Table

The Sent Reports table essentially contains the same fields as the Saved Reports table. However, they were distinguished for convenience reasons since each table allows for different operations. Once a saved report is sent, it is immediately removed from the Saved Reports table and added into the Sent Reports table.

### Modules and Interfaces

**Main Module:**

As mentioned in Chapter 4, the main module is the entry point of our application. It does not enclose much functionality, since all the main operations of our application our enclosed in the mini app.

This module consists of one activity. The activity contains two buttons, the START MANDOLAPP BUBBLE and the STOP MANDOLAPP.

When the START MANDOLAPP BUBBLE button is clicked a mini app is launched, as well as a service and a notification, only if they are not already launched. Details about the mini app, service and notification will be presented later on.

When the user clicks on the STOP MANDOLAPP bubble all operations of the Mandolapp application are stopped. This includes the mini app, notification and service.

******

Figure 27. Main Module Interface

**Mini App and Service:**

In order to provide the best possible experience to the user, we opted to give him a way of reporting without the need to leave the current application he is using. This was achieved using the Tooleap SDK, more details for which can be found in Chapter 2.

Using Tooleap, a mini app was created enclosing most of the application’s functionality. It consists of the bubble that is similar to the Facebook Messenger chat heads, that is continuously available and on top of other applications. By pressing on the bubble, a side screen pops out, that takes about the 2/3 of the device screen. The user can discard the mini app any time by dragging it in the “X” mark that will appear once the user presses and holds the bubble.

The mini app is launched once the user clicks the START MANDOLAPP BUBBLE button from the main module.

To further improve the experience of the user, a background service was also implemented. The background service has two basic jobs:

* To listen to which application is in the foreground
* To listen to Copy events

The service checks for which application is in the foreground every second. This is done so that when a user opens a specific application, our application will be able to provide him the proper guidelines on how to make the report for that specific application. For this specific feature to be implemented, after android API 19, a special permission is required so that the service can be able to get the foreground application. That permission is called Usage Access Permission. This permission has to be manually enabled by the user. Therefore, the first time that the main module launches, the user is redirected to the specific settings screen to enable the permission, before using the application.

Furthermore, since our application listens to Copy events for the reporting process, we wouldn’t want the application to capture copy events outside of one of the applications that the user can report to, as that would have a very negative impact on the user experience. The service checks the package name of the foreground application to recognize which application is in the foreground. Since there are many internet browsers available on the android play store we needed to include many possible package names to support different browsers. Currently, the browsers supported by the application are the android native browser, Google Chrome, Mozilla Firefox, Opera and UC Browser.

Every second, the service retrieves the foreground application. If that application is different than the previous application, meaning that it was just launched, and if the foreground application is one of the applications that the user can make a report from (Twitter, Facebook, browser) then the activity of the mini app is changed, and the appropriate activity is presented to the user.

The user is able to report a post by copying its link inside a specific application. For this functionality, the service must listen to Copy events. When it captures such an event, the copied content is checked to be a valid URL. If so, a new activity is loaded in the mini app for the user to report the post.

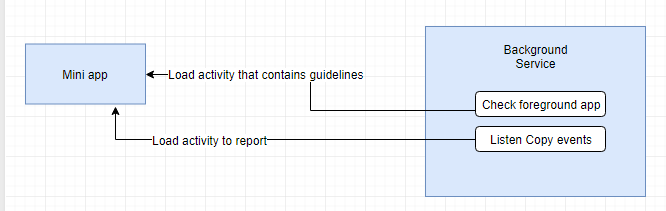


Figure 28. Background Service-Mini App Module Implementation

**Notification**

As already mentioned, the service checks for the foreground application each second. This can have an impact on the hardware resources of the devices, needing a lot of processing power and memory, along with possibly draining the battery life of the device faster.

Although no significant problem was expected, since modern android phones are powerful enough and already have a lot of processes running in the background, we decided to give the user a way to discard the mini app and be able to relaunch it from anywhere, without needing to visit the main application again.

This was done with the use of a persistent notification. The notification is launched when the user clicks the START MANDOLAPP BUBBLE button and cannot be discarded until the user clicks the STOP MANDOLAPP button.

The user can discard the mini app simply by dragging it in the “X” mark that will appear once the user clicks and holds the mini app bubble. When the mini app is discarded, the background service stops listening to any Copy events or the foreground application, thus freeing hardware resources.

In order to launch the bubble again all the user has to do is press on the notification that appears in the notification drawer of the device.

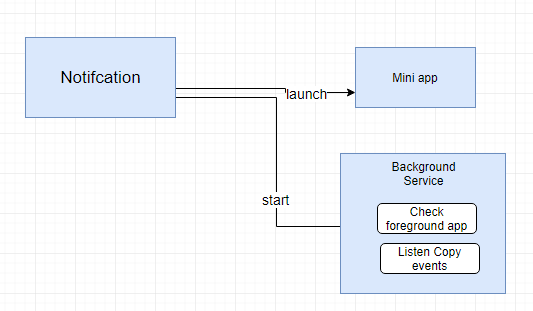


Figure 29. Notification Module Implementation

**Reporting Module:**

The reporting module is the entry point of the mini app, meaning that it’s the first screen a user sees when he opens the mini app and encloses the reporting functionality of the application.

It consists of 8 activities:

* ReportMain
* ReportBrowserFirst
* ReportBrowser
* ReportFacebookFirst
* ReportFacebook
* ReportPhoto
* ReportTwitterFirst
* ReportTwitter

ReportMain:

The first activity the user sees when he opens the mini app. It advices the user to open a social media application to make a report. It also provides a picture button for the user to press if he wants to report using a picture from his library.

If the picture button is pressed, a normal activity is started, redirecting the user to his picture library. From there the user can choose a picture just by clicking on it. When the user clicks on the picture, the picture library activity is closed, and the ReportPhoto activity is loaded in the mini app.

If instead of pressing on the picture button, the user opens one of the supported applications for report, the background service will load one of the ReportBrowserFirst, ReportTwitterFirst and ReportFacebookFirst activities in the mini app.

ReportTwitterFirst, ReportFacebookFirst, ReportBrowserFirst:

These activities are loaded in the mini app when a user opens the corresponding application. ReportTwitterFirst loads when Twitter is launched, ReportFacebookFirst is loaded when Facebook is launched and ReportBrowserFirst is loaded when an internet browser is launched.

The ReportTwitterFirst and ReportFacebookFirst activities contain information on how the user can copy a link of a post by providing a moving picture (gif) that shows the process.

The ReportBrowserFirst activity just tells the user to copy the link of content on the web. Specific guidelines could not be given, since the user can copy any content that has a public link.

When the user copies a link, the background service loads the corresponding activity in the mini app.

ReportTwitter:

The activity loaded when the link of a tweet is copied. The activity loads the tweet copied using Twitter Kit and the Twitter Developer API.

For this to happen, the id of the tweet is isolated and the tweet is requested from the Twitter API using its id. When the tweet is returned, it is embedded in the activity layout. For the retrieval of tweets, a Twitter application was created, named Mandolapp. Although the Twitter application credentials are used for OAuth2 validation before retrieving the tweets, the validation is done as a guest authentication. This means that each user can request up to 180 tweets per 15 minutes. Even if the user exceeds that amount, a normal android WebView replaces the embedded tweet and the tweet is loaded as html content, so the user can always preview the tweet.

ReportFacebook:

The activity is launched when the link of a public Facebook post is copied. The embedding of the post is an easier process than that of a tweet. The link of the post is sent to the Facebook API which returns the post in web content, that is displayed inside a WebView.

ReportBrowser:

The activity is launched when a link is copied from an internet browser application. The embedding of the content is done using a WebView, since the user can copy the link of any content in a browser.

ReportPhoto:

The activity is launched when the user selects a picture from the picture library that is launched when the user clicks the picture report button from the ReportMain activity. The picture is loaded so the user can preview the picture before making the repot.

The activities that load for the user to make a report (ReportTwitter, ReportFacebook, ReportBrowser, ReportPhoto) also provide a form to the user to give details about the report, exactly as described in the Chapter 3 and Chapter 4.

When a user chooses to save a report, the report is saved in the Saved Reports table of the SQLite database. If he chooses to send the report, the report is sent to the REST API we developed and it is also saved in the Sent Reports table of the SQLite database.

**Reporting Module Implementation:**

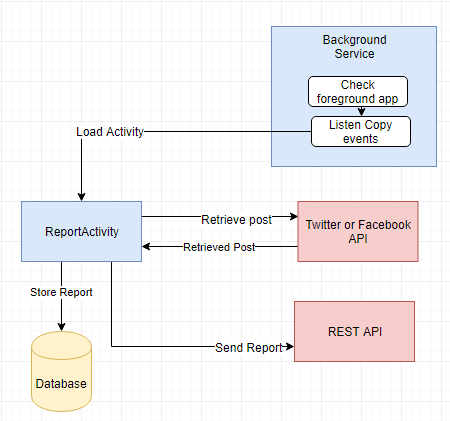


Figure 30. Report Module Implementation

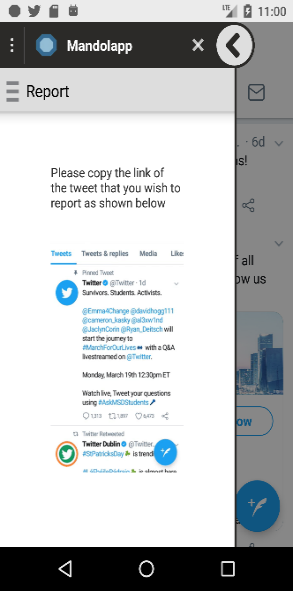
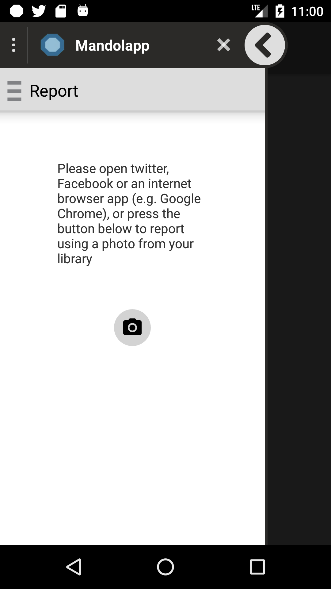
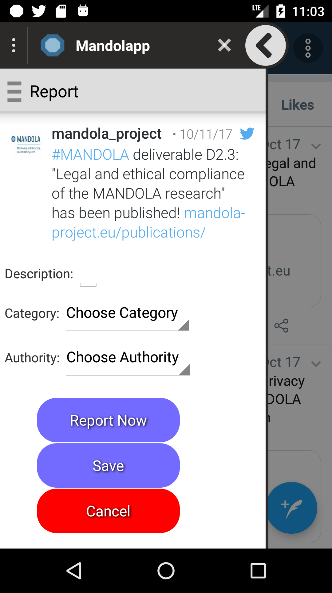
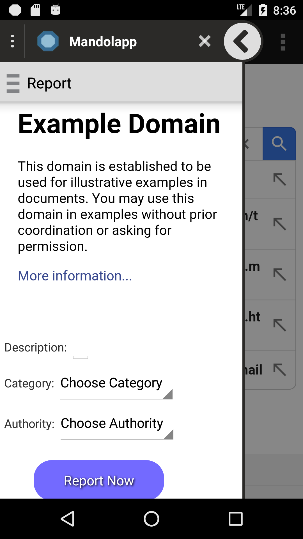
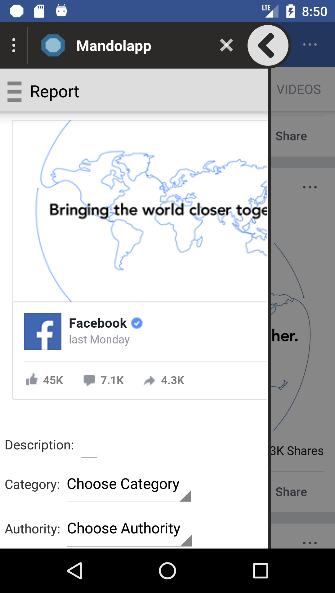
**********Reporting Module Interfaces:**

Figure . Reporting Module Interface

**Saved Reports Module:**

As the previously stated the aim of this Module is to provide users with the ability to view saved reports as well as further edit them, delete them, or send them.

The Saved Reports Module consists of 4 activities:

* SavedReports
* SavedReportsBrowser
* SavedReportsFacebook
* SavedReportsTwitter

SavedReports:

The SavedReports activity, is the first activity the user sees when he navigates to the Saved Reports module.

The activity retrieves all the saved reports from the Saved Reports table of the database and presents in a List View, in a friendly user interface and sorted from the newest to the oldest. The user can delete a report by pressing a button provided for each report and can also filter the reports presented to him.

The user can filter the reports by the application they were made from (Twitter, Facebook, browser). The user can also search for reports, using the date they were saved or their description, while the filtering based on application might still hold.

For the filtering and sorting SQLite operations are used, to increase speed and reliability. Since, the user can enter any string in the search box, we also made sure that our database is protected against SQL injection.

When the user clicks on a specific report from the List View the corresponding activity is loaded and the chose report will be passed to it by this activity.

SavedReportsBrowser, SavedReportsFacebook & SavedReportsTwitter:

When the user clicks on a specific report from the SavedReports activity the corresponding activity of these 3 is loaded.

Just like in the reporting module, these 3 activities will retrieve the post or content to be reported, using the Twitter API, Facebook API, or a simple WebView in the case of a browser report.

They present any user details the user has already chosen and allow him to further edit them. By pressing on the Delete button, the report is removed from the database, while clicking on the Report Button will result in the report being sent to the REST API and the report being deleted from the Saved Reports table and added to the Sent Reports table.

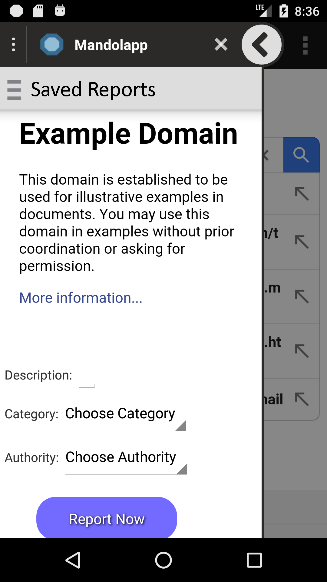
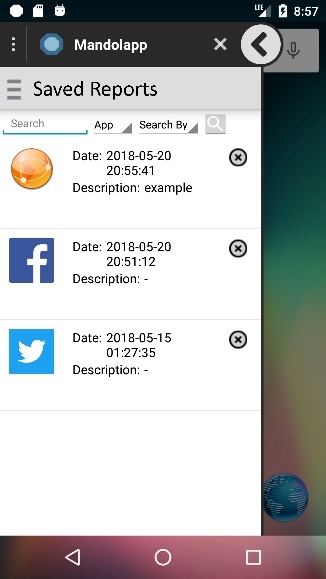
******

Figure . Saved Reports Interface

**Report History Module**

The application provides to the user the ability to view his report history. This module consists of 5 activities:

* ReportHistory
* ReportHistoryBrowser
* ReportHistoryPhoto
* ReportHistoryTwitter
* ReportHistoryFacebook

ReportHistory

The ReportHistory activity is the main entry point and the first activity to load in the Report History Module. Much like the SavedReports activity it retrieves the reports from the Sent Reports table and presents them in a List View. The activity provides the same options for filtering, searching and deleting a report as the SavedReports activity.

When a report is clicked the appropriate activity is launched.

ReportHistoryBrowser, ReportHistoryPhoto, ReportHistoryTwitter, ReportHistoryFacebook:

When the user clicks on a report in the ReportHistory activity, the appropriate activity from these 4 is launched, based on the application the report was made from.

The post or content of the reports is loaded just like in previous activities and details about the reports are presented to the user.

By clicking on the Clear button, the report is deleted from the Sent Reports table.

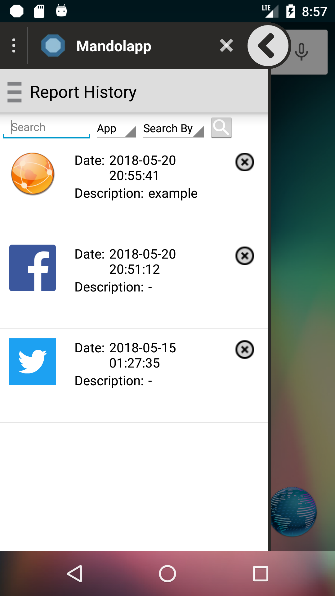
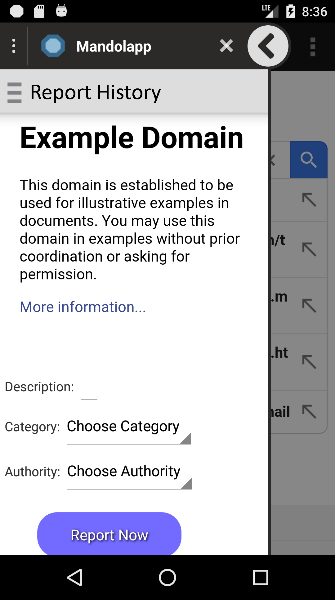
******

Figure . Report History Module Interface

**About Module:**

The about module provides some information about the application and its purpose as well as the mandola project. It consists of the About activity.

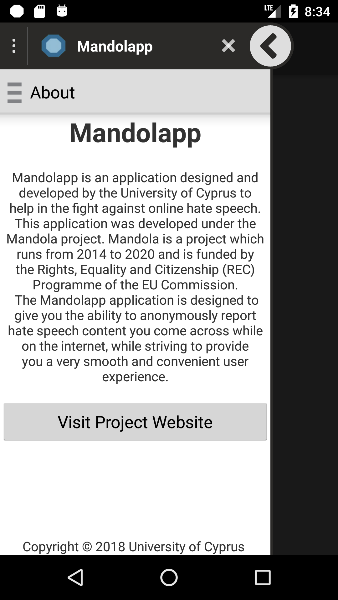


Figure . About Module Interface

**Statistics Module:**

This module provides statistics regarding the report history of the user. It consists of the Statistics activity.

The statistics that the user is able to see are the following:

* Number of reports per category
* Number of reports per application (Twitter, Facebook, Browser, Picture)
* Number of reports per category for Twitter reports
* Number of reports per category for Facebook posts
* Number of reports per category for browser posts
* Number of reports per category for picture reports

These statistics are presented in the form of pie charts.

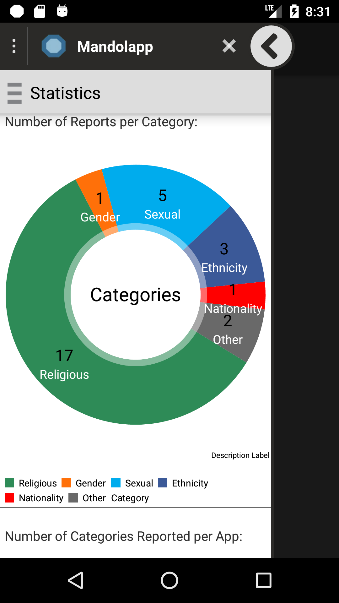
******

Figure 35. Statistics Module Interface

**Settings Module:**

The settings module provides the user the ability to customize his experience while using the application. Is consists of the Settings activity.

The user has the option to turn on or off the Report Guidelines settings using a switch. Report Guidelines are basically the moving pictures (gif) that appear in the Reporting Module to show the user how to copy a link of a post.

The user can also choose some default authorities to be preselected, every time he attempts to make a report for his own convenience. The settings are saved using the SharedPreferences interface of the android API so that they are accessible by all our activities, but only from our application.

The user can also choose to delete all saved reports or all of the report history from their respective database tables, by clicking on the corresponding buttons that are provided to him.

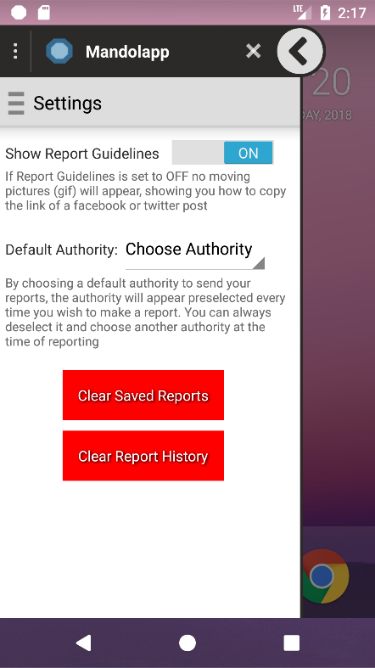
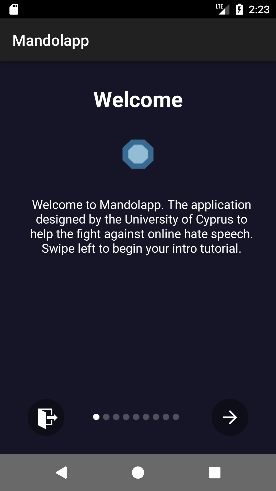
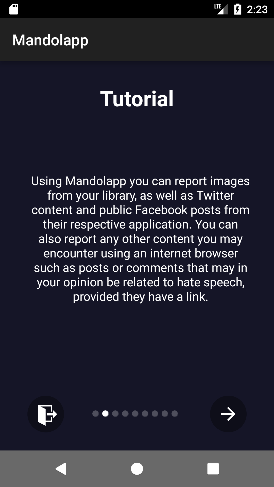
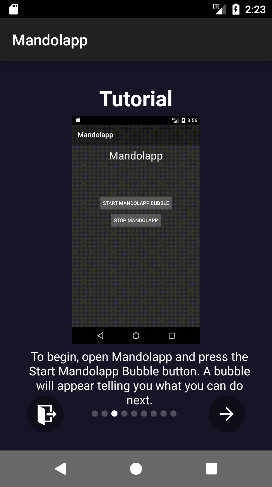
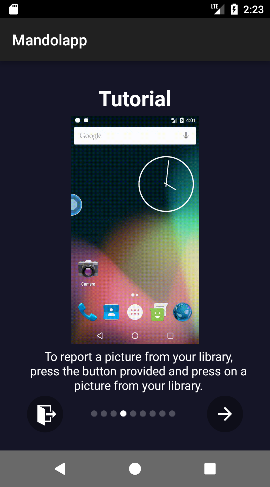
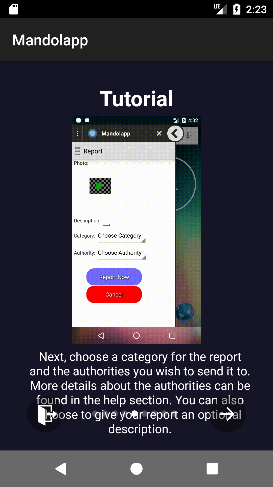


Figure 36. Settings Module Interface

**Introduction Module:**

The introduction module is launched the very first time our application is launched, to provide to the user a small tutorial on how to use the basic features of our application. Although not in the design mock-ups, it was discovered that users had some trouble understanding how to start using the application. Therefore, the need for a short introduction was established. The user can discard the tutorial at any time and access it again in the help section.

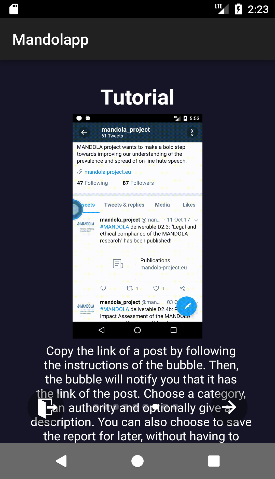
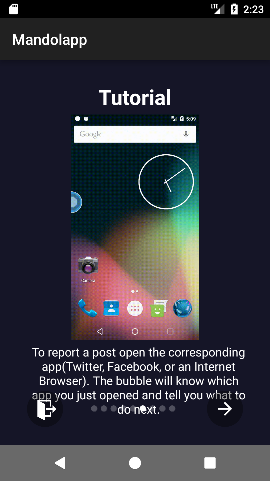
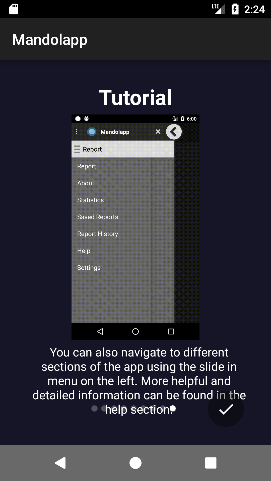
  

Figure . Introduction Module Interface

**Help Module:**

The help module provides basic information to the user about how to make a report for each application, description about each category and authority, as well as an FAQ (Frequently Asked Questions) section. It consists of 9 activities:

* Help
* HelpFAQ
* HelpReportInstructions
* HelpReportInstructionsTwitter
* HelpReportInstructionsFacebook
* HelpReportInstructionsBrowser
* HelpReportInstructionsPicture
* HelpCategories
* HelpAuthorities

Help:

The help activity is the entry point of the help module. From there, the user can navigate to the HelpFAQ or HelpReportInstructions.

HelpFAQ:

This activity contains the Frequently Asked Questions and their answers.

HelpReportInstructions:

This activity gives the user the ability to choose to see reporting instructions about Twitter, Facebook, internet browser and picture reports, details about the different categories, details about the different authorities and to re-watch the introduction tutorial.

HelpReportInstructionsTwitter, HelpReportInstructionsFacebook, HelpReportInstructionsBrowser, HelpReportInstructionsPicture:

Activities containing step by step instructions about the report process of the corresponding application.

HelpCategories:

This activity contains details about the different kinds of hate speech categories.

HelpAuthorities:

This activity contains details about the different authorities.

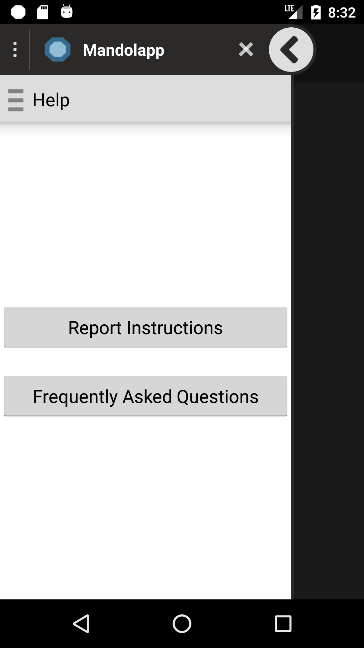
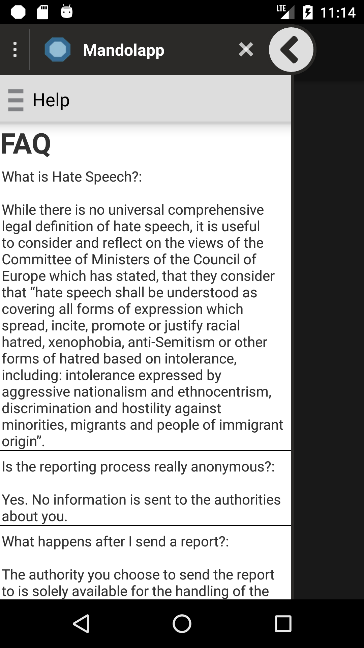
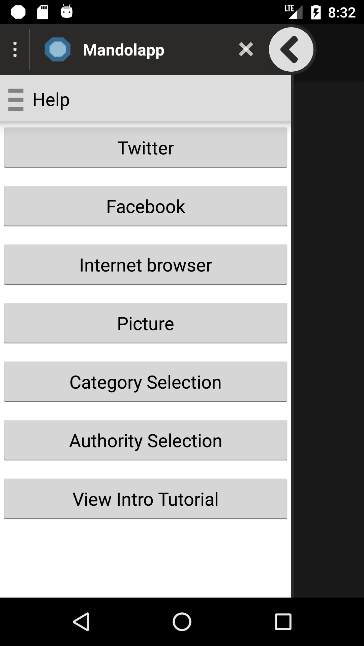
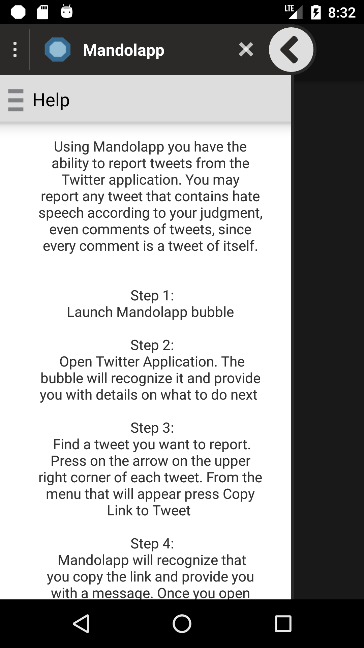
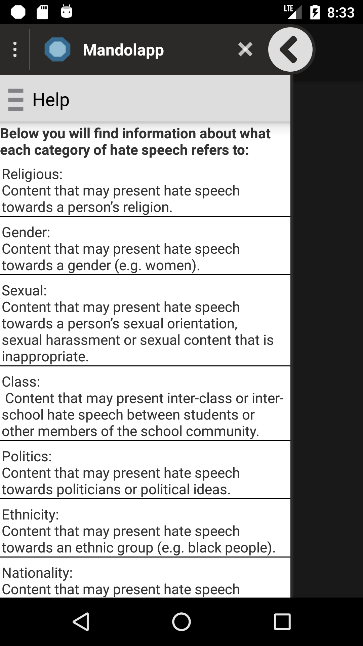
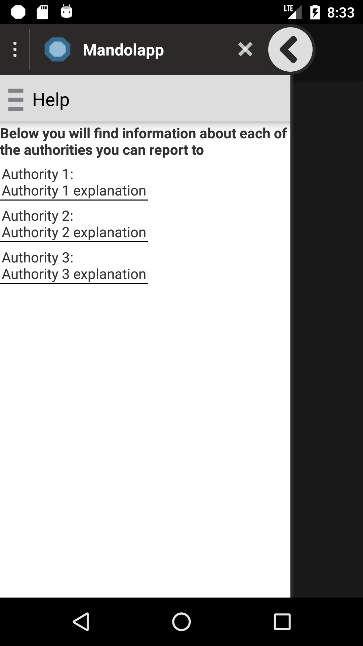


Figure . Help Module Interface

### Compatibility

**Backwards Compatibility**

As stated in the non-functional requirements section of Chapter 3, the application needs to be backward compatible, supporting at least android API level 19. The implementation of the application allows that. The application could be compatible in lower versions of the android API down to level 14, however its operability is not guaranteed since under API level 19 significant problems were detected.

**Other Compatibility:**

As stated earlier, in order for the background service of our application to be able to get the foreground application, the Usage Access Permission is required. However, it was discovered that certain models of android phones, operating certain android versions, don’t have this permission, either by fault of the manufacturer or by deliberate removal.

Although not officially confirmed, a list of device models and the respective android version that lack the Usage Access Permission is the following:

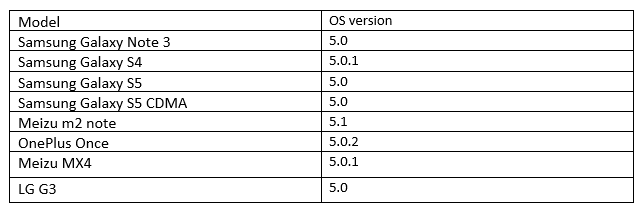


Figure 39. Smartphone Models & Android versions with no Usage Access Permission [19]

It should be noted that the Usage Access Permission is available on some different android versions of the same devices.

## REST API

Along the smartphone application, a REST API was developed, to receive the reports sent by our application.

The REST API was developed using Java Spark, more details for which can be found in Chapter 2.

Java Spark was chosen as the appropriate framework, since it is a very lightweight framework, unlike other java frameworks like Spring MVC for example, while it provides all the functionality that we need. Another important aspect is the fact that it can be used with the Java programming language, so the transition between the development of the application and the API would be smoother.

### Database

A relational MySQL database was created, to store the reports from the application. The database contains 4 tables:

* TwitterReports
* FacebookReports
* BrowserReports
* PictureReports

TwitterReports, FacebookReports & BrowserReports:

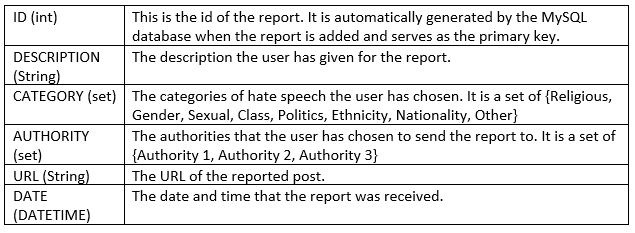


Figure 40. TwitterReports, FacebookReports & BrowserReports REST API tables

PictureReports:

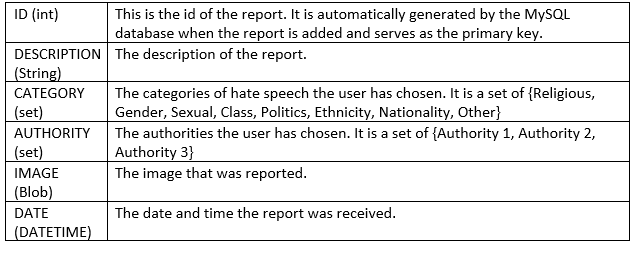


Figure 41. Picture Reports REST API table

Each table corresponds to the corresponding application that the report was made from. Unlike the smartphone application database, a table for each application type was developed since this API is not concerned with extensibility issues.

### API – application communication:

The communication between the API and the smartphone application is done using HTTP POST messages.

In order to send a picture to the API the application will send an HTTP POST request to the URL ‘http://hostname/addPictureReport’. The data of the POST request will consist of the report attributes.

Example data: {“id”:”0”,”App”:”picture”,”Image”:”Base64EncodedImage”,”Description”:”ReportDescription”,”Category”:”Category1, Category 2”,”Authority”:”Authority 1, Authority2”}

In order to send other types of reports to the API the application sends an HTTP POST request to the URL ‘http://hostname/addReport’. The data of the POST request, once again consists of the report attributes.

Example data:

{“id”:”0”,”App”:”Twitter”,”Url”:”http://hostname”,”Description”:”ReportDesciption”:”Category”:”Category 1, Category2”,”Authority”:”Authority 1, Authority2”}

Once the HTTP POST request is received by the API, the report will be saved into the appropriate table in the database. If the storing of the report is successful the API returns the success code 200, otherwise the code 500 is returned.

# Chapter 6

**Evaluation**

6.1 Performance Metrics

6.2 Beta Testing

The evaluation phase is a very important phase in the development life-cycle of our application. This phase involves the evaluation of the performance as well as the efficiency and usability of the application.

The application is designed around the concept of providing to the user the best possible user experience. The processes of our application must be easy and convenient for the user to use while the application itself must be fast and as lightweight as possible in regards to the hardware resources usage.

## Performance Metrics

Since the main idea behind the application is the convenience of the user, the performance of the application must be tested. By testing the performance of the application, we can make sure that the application is fast enough to meet the requirements of today’s fast paced users.

Additionally, the hardware resource usage of the application must be tested. The application encloses some complex functionality, while it is always running in the background, utilizing hardware resources.

By testing the resource usage of the application, we must make sure that the application doesn’t sacrifice hardware power and battery life in the sake of being more “convenient”, since such situations can truly downgrade the convenience and usability of our application.

For the performance metrics the android profiler was used. The android profiler can provide real-time data for our application’s CPU and memory usage. [20]

For the execution of the performance metrics an android emulator was used. The application was installed on the android emulator using android studio and a set of use cases were carried out to test the performance of the application.

The android emulator that was used had the following hardware specifications:

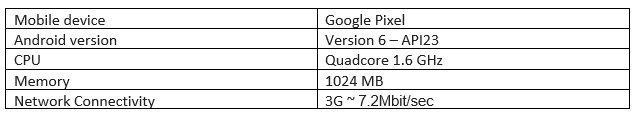


Figure 42. Performance Metrics Device Specifications

By executing certain tasks on the android emulator, the android profiler is able to show the CPU and memory usage of the device in real time. The following metrics were calculated in the experiments to follow:

* CPU Usage:

A very important metric for any application is the CPU usage. The CPU directly reflects the effect of the application on the battery life, as well as on the overall performance and responsiveness of the device.

For this performance metric, we used the android profile CPU usage monitor.

* Memory Usage:

Another important metric of our system is the memory usage. Memory usage can help us identify memory leaks in our system as well as memory intensive tasks which might have an impact on the responsiveness of our application or device. For this metric the Memory profiler of the Android profiler was used.

* Time to Load Content or Send Reports:

Since the application is developed around the idea of providing a fast and convenient experience to user, we must make sure that this experience doesn’t suffer from slow performance. This metric was carried out by adding code to important parts of the system to see how long it took for them to be executed.

**Experiment 1: Load Tweet, Facebook post, Internet browser content & Picture**

For this performance metric, our application bubble was first launched. Then we followed the procedure for making a report, by opening each of the social media applications (Twitter, Facebook, Internet browser) and copying the link of a content or post to be reported.

Once the link is copied and the bubble pressed, our application will load and present the copied content for the user to preview before making the report. In this experiment we will measure the CPU and memory usage that our application consumes while loading that content as well as the time it takes to load it.

As for the pictures, the camera button of the main report screen of our application was pressed to reveal the picture library of the device. The appropriate picture was clicked and the application loaded the picture for preview before reporting. For the loading of the pictures, pictures of resolution 1280 x 960 and sized 221KB were used.

For each application the test was carried out 5 times to get the average values of all metrics.

CPU usage while loading content from applications:

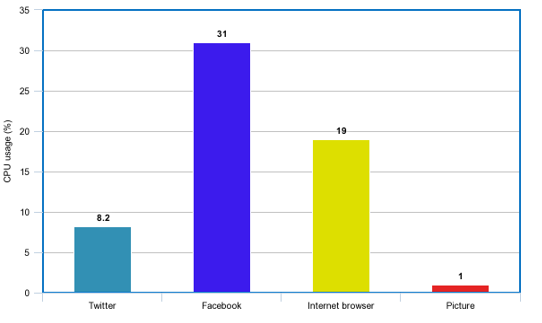


Figure . CPU Usage While Loading Content

As we can see in figure 43 in order for our application to load content from social media sites, a considerable amount of CPU power is used. However, this CPU usage takes place for some seconds only, while the content is being loaded and therefore isn’t expected to have an impact on the battery life of the device.

Memory usage while loading content from applications:

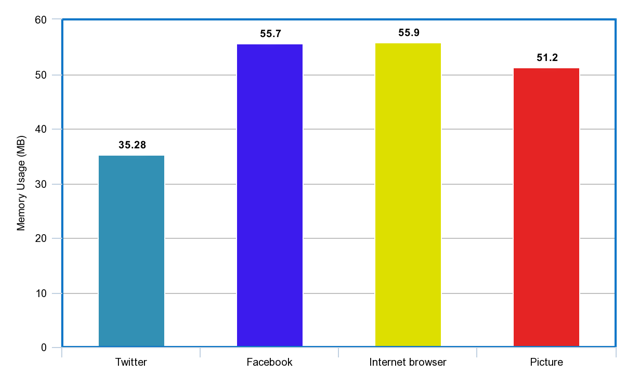


Figure 44. Memory Usage While Loading Content

In figure 44 we can see the memory used by our application (in MB) when the content of a social media application is presented. Smartphone devices nowadays have some GB of memory, so our application isn’t expected to affect the responsiveness of the device.

Time to load each kind of content:

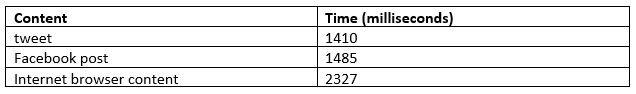


Figure 45. Time to Load Each Kind of Content

The loading of social media contents doesn’t longer than 3 seconds as we can see and therefore we don’t expect to have any negative impact on the user experience.

**Experiment 2: Send Reports**

As stated previously, our application was developed around the concept of providing the best possible user experience. Slow reporting is certainly not a characteristic of a good user experience, we must therefore make sure that the sending of reports is not a time-wasting process.

For this experiment, each kind we sent a report of each kind (Twitter, Facebook, Internet browser and Picture) to our API and measured the time it took for our application to receive an answer from it.

Time to send each kind of report:

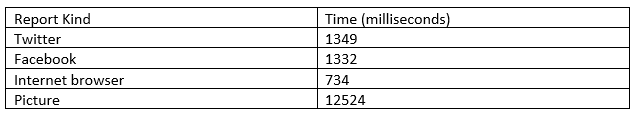


Figure 46. Time to Send Each Kind of Report

The sending of reports seems to be a relative fast process, since all kinds of reports except the picture reports don’t take longer than 2 seconds on average to be reported. However, we can see that picture reports take about 12 seconds to be reported. This is mostly due to the size of the pictures, since the pictures used in this experiment are relatively large in size.

**Experiment 3: Browsing while the bubble is active**

This experiment aims at seeing the effect that the application has on the battery life and memory just by being active. The reason behind the need for this experiment is the fact that our application always listens for either the launch of a social media application that is supported by our application for reporting, or the copy of a link from one of those applications. Since the application is aimed at being convenient enough for the user to keep it continuously open, while the application always listens for one the events described earlier, it is expected that this will have an effect on the battery life of the device.

To carry out this experience the bubble was launched from the main application, while a simple browsing through different features of the device took place.

To further understand the CPU usage values for our application, we decided to compare our application to another, to see the differences. We decided that this application would be Google Chrome, an application that although is surely not the most lightweight, is an application that the user expects to always run in the background and perform heavy tasks with, without any considerate effect on the performance or battery life of the device.

The metrics were performed on the process assumed to be the most important in the device’s battery life: the background service, which is the service to listen to app launches and copy events.

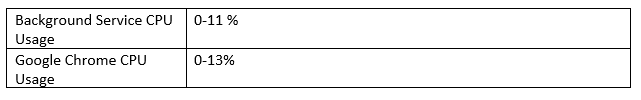


Figure 47. Background Service CPU Usage

The CPU usage is expressed in a range from the smallest number encountered while browsing to the largest, over a period of 10 minutes of browsing through different features. The CPU usage is normally expected to have small changes in its value over a small period of time.

As we can see our application did pretty well, reaching a maximum CPU usage at 11% when a social media app was launched. By contrast, Google Chrome had a maximum CPU usage of 13% while running in the background. This means that the background service of our application can run in the background seamlessly without any considerably effects in battery life just like the Google Chrome application would be expected to run.

Furthermore, the bubble of our application can be discarded at any time and relaunched by the notification in the notification drawer, a functionality added to improve battery life. When the bubble is discarded, the background service doesn’t listen for app launches or copy events and therefore the CPU usage is at 0%.

As in the CPU usage metric, the application’s memory usage was tested while the background service was running in the background, while Google Chrome was also running in the background.



Figure 48. Background Service Memory Usage

As we see once again, our application seems to be enough lightweight to not cause any considerable problems while running in the background to the device’s performance.

## Beta Testing

After measuring the performance of the application in numbers, the application must be tested by real-world users, in order to be able to truly test its convenience.

Since the application targets every day users, it was decided that normal every day users must participate in the beta testing process, without the need for any specific computer science knowledge. The only requirement for the participating users would be that that would have to have access to a smartphone device and be familiar with the use of social media.

We managed to gather 15 participants. Out of the 15, 8 are current university students, 3 are recent university graduates, 2 are bank employees, 1 is a Special Advisor for school and family violence in government schools and 1 is self-employed. The age range is between 19 to 40 years old. Although this range is not huge, it is sufficient for our application to be tested by teenagers, young adults and middle-aged people.

The beta testing lasted for a period of two weeks. The participants were asked to use the application as much as possible, make reports, save reports and perform every possible operation the application provides. The participants were also asked to particularly report twitter posts, for a statistical analysis to be made later on.

### Questionnaire

After the period of 2 weeks was over, a questionnaire was created, using Google Forms, so that the users could give their feedbacks.

The questionnaire includes 3 types of questions:

* Multiple choice: The multiple-choice questions are targeted towards finding out about specific details of the users’ experience. Each multiple-choice question consists of 5 possible answers.
* Range (e.g. 1-5): Range questions were used to find out about how strongly or not the users felt about the answer to a specific question that they were asked. A range of 1-5 was used for most questions, while 1 question used a range of 1-10. Some of these questions were also used for the exporting of frequently asked questions.
* Open answer questions: These questions required the user to write down their opinion about the question. Such questions were used to get information from the users on topics that could not simply be answered with a multiple-choice answer. These questions were optional.

Question 1 - What was the speed of the installation process like?

This question is targeted towards getting feedback about the installation process of our application. Although the package size is small and the installation process per se is fast, the user needs some time to get started using the application since he must go through the introduction tutorial and also enable the Usage Access Settings. This way we can see if and how much of a negative impact this is for our application.

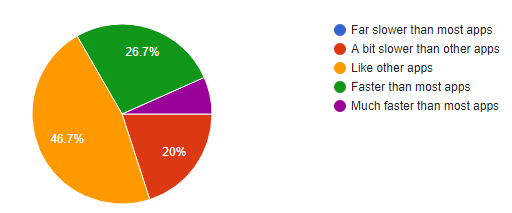


Figure 49. Questionnaire Question 1 Answers

As we can see from the pie chart above, the majority of the users (46.7%) thought that the installation process was like applications. Few people thought that the it was faster or slower than other application, while just one user thought that it was much faster.

This means that the introduction tutorial or the enabling of the Usage Access Permission didn’t have that much of negative impact on the user experience, just as he had hoped.

Question 2 - How easy was the app to use?

During the earliest stage of the beta testing phase, it was noticed that although the application offered a lot of convenient features to the users, the first couple of users that got the application could not easily figure out how to use it. Therefore, it was decided that an introduction tutorial would be created that would launch on the very first time the application is launched. This question will help us figure out to what extend the problem was solved, after the introduction of tutorial.

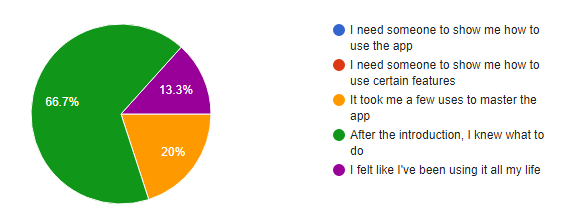


Figure 50. Questionnaire Question 2 Answers

In the pie chart above, we can clearly see that the vast majority of users (66.7%) found the introduction very useful. Fewer people still struggled a bit with the use of the application while even fewer people were very confident, answering that they felt they were using it all their life.

This means that the introduction tutorial that was introduces into our application really made a difference in the understanding of the operation of the application. Therefore, we can assume that the majority of the users that would perhaps download this application from the Google Play Store at a later stage would know how to start using it.

Question 3 -How clear was the purpose of the application?

We would like to see whether the users had truly understood the purpose of the application and why the need to use it.

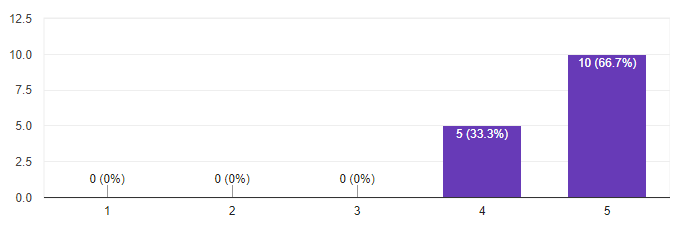


Figure 51. Questionnaire Question 3 Answers

The minimum number 1, stands for Not Clear at all, while 5 stands for Very Clear.

In the above bar chart, we can see that the introduction did a good job explaining the purpose of the application and perhaps there is no further need in explaining it further.

Question 4 - How clear is to you, what hate speech is?

Since many countries don’t have a solid definition of what hate speech is, is only normal that many people would have trouble understanding what hate speech is.

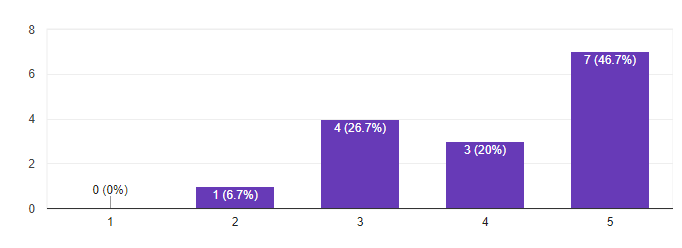


Figure 52. Questionnaire Question 4 Answer

The minimum number 1, stands for Not Clear at all, while 5 stands for Very Clear.

Although we can see that to the majority of users (46.7% or 7/15) it is very clear what hate speech is, a total of 33.4% of the users answered between number 2 and 3. That means that about a 3rd of our users weren’t very confident on what hate speech. Therefore, a frequently asked question was added, to help the users better understand.

Question 5 - How clear were to you, the kind of posts you can report for each kind of application (Twitter, Facebook, Internet browser, Photos library)?

Our application supports a variety of different report types. For each report type the user can report different kinds of posts. For example, he can report all tweets and even comments in the Twitter app, only public posts in the Facebook app and any content with a publicly accessible link in the internet browser app. Although guidelines were given to the users, we wanted to make sure that those guidelines were adequate for the understanding of the different kinds of reports.

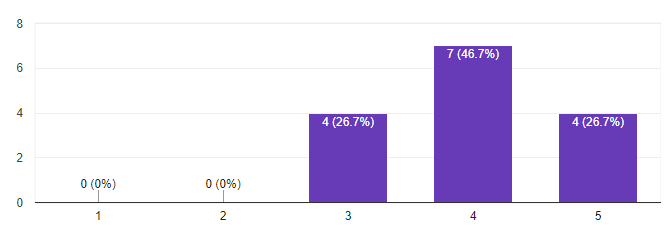


Figure 53. Questionnaire Question 5 Answer

The minimum number 1, stands for Not Clear at all, while 5 stands for Very Clear.

It seems that most users (46.7%) that answered number 4 were a bit confident, while 26.7% of people seem to be very confident and 26.7% not very confident. Consequently, a detailed section was added in the help section, containing information on how to make a report for each application and what exactly can be reported.

Question 6 - How clear was to you that the reporting procedure is anonymous?

This questions just aims at finding out if the users could recognize the fact that the reporting procedure was anonymous, or if the application needed to further point it out.

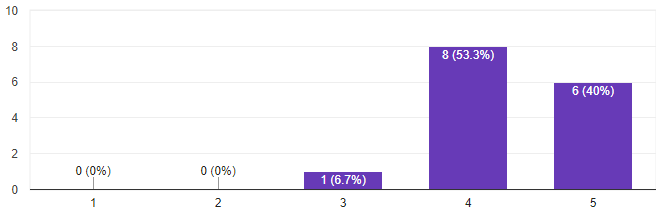


Figure 54. Questionnaire Question 6 Answers

The minimum number 1, stands for Not Clear at all, while 5 stands for Very Clear.

Most of our users seem to understand that the reporting procedure is anonymous. However, 1 person answered number 3. It was decided that a frequently asked question would be add to inform the users on their anonymity in case they are not sure.

Question 7 - Did the app help solve your problem/ achieve your goal of reporting hate speech?

Since there is already a number of available applications for the reporting of hate speech, as well as the individual reporting platforms of each social media application, we would like to know if the users thought that the application helped them in the reporting of hate speech, or if they believe that they could have reported hate speech more easily, in another way.

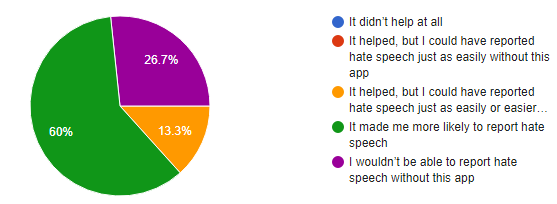


Figure 55. Questionnaire Question 7 Answer

The vast majority of the participants (60%) thought that the application made them more likely to report hate speech while some answered that they would only report hate speech with this application and fewer people answered that they could just use another application just as easily. Given the large number of hate speech reporting applications in the market and the individual platforms of each social media app, it seems that our application succeeded in offering something different to the users that made the reporting process more convenient than most apps.

Question 8 - How convenient did you find the reporting process of the application?

Our application is targeted towards providing the best and most convenient experience to the user. Hence, this question aims at finding how convenient the users found our application.

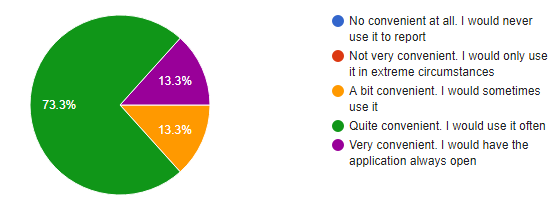


Figure 56. Questionnaire Question 8 Answer

The huge majority of the participants seem to have found the reporting process of the application very convenient. Hence, we have succeeded in providing a truly convenient reporting process to the users.

Question 9 - Which functions didn't work as expected?

This question clearly aims at finding bugs in our application that the users might have identified.

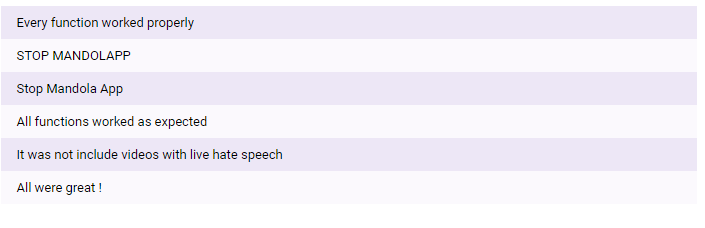


Figure 57. Questionnaire Question 9 Answers

Some users reported that the STOP MANDOLAPP button from the main screen of the main application was not working properly. The problem was found and corrected.

Question 10 - What features would you add?

Aims at getting the opinion of the users about further work that could be done on the application.

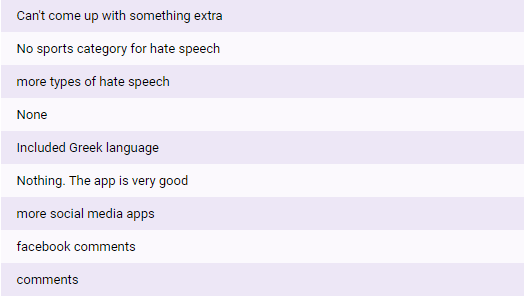


Figure 58. Questionnaire Question 10 Answers

Some people answered that they would like to see more hate speech categories added, while some people said that they would like to be able to report content from more social media applications, as well as social media comments, an option that so far is only available in the Twitter application.

All the suggestions could be taken into account for future work that may be done in the project.

Question 11 - Given the app’s purpose to help you report hate speech content, why wouldn’t you use this app again?

This question aims at breaking down our application. We want to test the application by asking users to think of reasons they wouldn’t want to use it again.

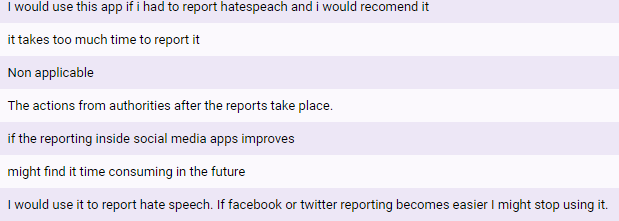


Figure 59. Questionnaire Question 11 Answers

We can see that 1 user didn’t find the reporting process fast enough to continue using the application consistently, while some people answered that they could stop using the application if the reporting process that each individual social media platform provides, becomes more convenient.

Question 12 - What other questions were raised to you during the use of the application, if any?

The questions at finding possible user questions that weren’t addressed by the application, to be added to frequently asked questions.

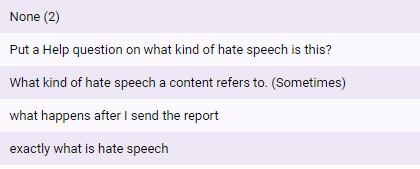


Figure 60. Questionnaire Question 12 Answers

The answers from this question were taken into account and were answered in the help and FAQ section.

Question 13 - How likely are you to recommend this application to a friend?

This question aims at truly finding out about the experience of the user, since users who are satisfied with the application would most likely recommend it to a friend. Using this question, we will calculate the Net Promoter Score (NPS) of our application and help us see how easily or not the application would spread in the user community. The answer of the question ranges from 1-10. People who respond with a 0 to 6 are called “Detractors”, with 6-8 are “Passives” and with 9-10 are “Promoters”. [21]

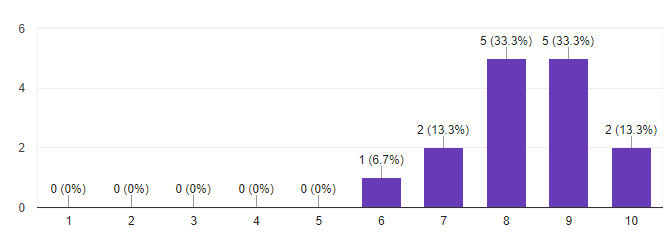


Figure 61. Questionnaire Question 13 Answers

The NPS score of our application is 8 which is a passive score. It seems that a 3rd of the users answered number 8, while a 3rd of the users answered number 9. The NPS score indicates that our application would not spread very fast in the smartphone market, although the fact the score is in the highest end of the passive scores range, gives us confidence.

# Chapter 7

**Reports Statistic Analysis**

7.1 Methodology

7.2 Statistics

Under the scope of this project, we decided to gather real hate speech reports and provide a simple statistical analysis on them to extract import statistics and information about hate speech in social media. It was decided that statistics would only be extracted for twitter reports, since the Mandola endpoint which will scan a report’s text is specifically designed for tweets, while the retrieval of hate speech content from other types of reports is a far more complicated process. Additionally, the reports were asked to be in English, since the endpoint can so far recognize only the English language.

## Methodology

**Report Gathering**

The gathering of reports was done during the beta testing period. Participants of the beta testing phase reported tweets, among other content, containing real hate speech.

For this to be able to happen, the REST API we created was uploaded to an AWS EC2 instance, so that it could be on a public server that all the users would have access to. An AWS EC2 instance, is basically a virtual server on Amazon’s Elastic Compute Cloud(EC2) running on the Amazon Web Services Infrastructure (AWS). [22]

**Tweet Text Retrieval**

Since twitter reports don’t contain the text of the tweet, it should be retrieved from the REST API. Using similar methods as the application the REST API would request a tweet from the Twitter Developer API based on the id of the tweet, which was extracted from its URL.

For the connection with the Twitter API, Oath2 app-only authentication was used, using the Mandolapp Twitter application we created as a credential. This resulted in being able to request 900 tweets per 15 minutes, far more than we were able to gather.

**Hate Speech Recognition**

After the retrieval of the tweets from the Twitter API, the text of each tweet was sent to the Mandola endpoint to recognize any hate speech content.

The Mandola endpoint is an API created by the University of Cyprus under the scope of the Mandola project [http://mandola-project.eu/].

The mandola endpoint receives HTTP POST requests where the text of a tweet is in the request body and returns the percentage of hate speech found.

Example data: {“text”:” tweet text”}

Example response: “40% Hate Speech”

**Statistical Analysis**

After getting the hate speech percentage of each tweet, tweets that were found to have 0 hate speech by the Mandola endpoint were excluded from the statistical analysis.

The following statistics were extracted:

* Reports per Hate Speech percentage
* Reports per Category
* Reports per Country
* Reports per Time Period
* Number of Reports for each Category per Time Period

## Statistics

During the beta testing period we managed to collect a total of 115 twitter reports from the 15 participants that took part.

**Removed Tweets**

Out of the 115 tweets, 25 tweets were either removed or the Twitter was banned, probably because the Twitter platform received complaints about them as well. Therefore, only 85 out of the 115 tweets could be retrieved from the Twitter API.

**Reports per Hate Speech Percentage**

The 85 remaining tweets were sent to the Mandola API to retrieve their hate speech percentage. For this statistic the Hate Speech percentage was divided into different ranges.

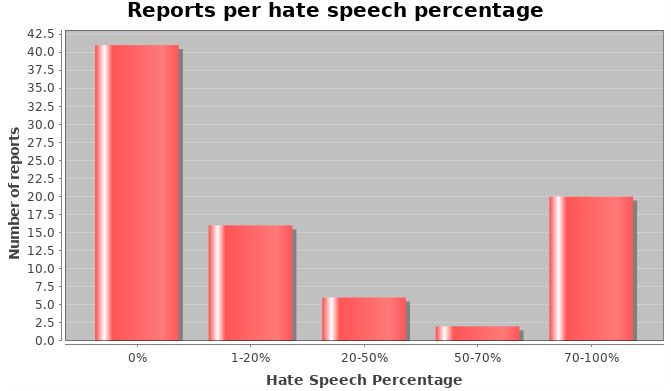


Figure 62. Reports per Hate Speech Percentage Bar Chart

As we can see the majority of the reports were recognized as hate speech free. Particularly, 41 reports have exactly 0% of hate speech content. In the same time a number of 20 reports were recognized as containing 70-100% hate speech content, while the other 25 reports fall in the range of 1-70% (non-inclusive) hate speech.

It was decided that for the rest of the statistical analysis process only the reports that were recognized as containing hate speech by the Mandola endpoint would be used. That leaves us with a number of 44 reports.

**Reports per Category**

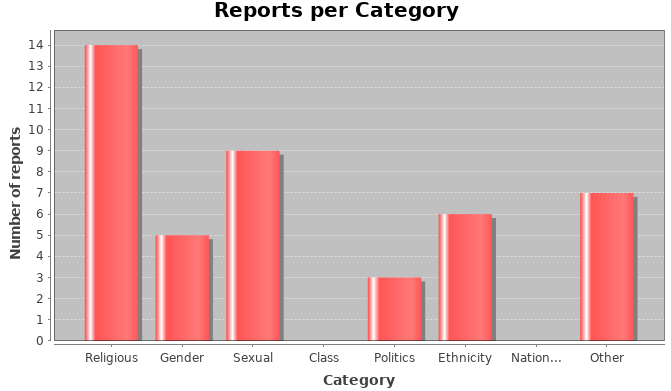


Figure 63. Reports per Category Bar Chart

In figure 56 we are able to see the number of the reports we confirmed as containing hate speech content, per Category. Since the user can tag more than one category for each report, each report can contain many categories. That’s why in the above bar chart the total number of reports is larger than 44 (number of reports used for the statistical analysis).

As we can see, the Religious category is tagged in 14 reports, making it the most frequent in our data set. However, almost every category appears in our dataset, except for the Class and Nationality categories.

Such statistics can help further understand the nature of hate speech as well as help tackle each hate speech category independently.

**Reports per Country**

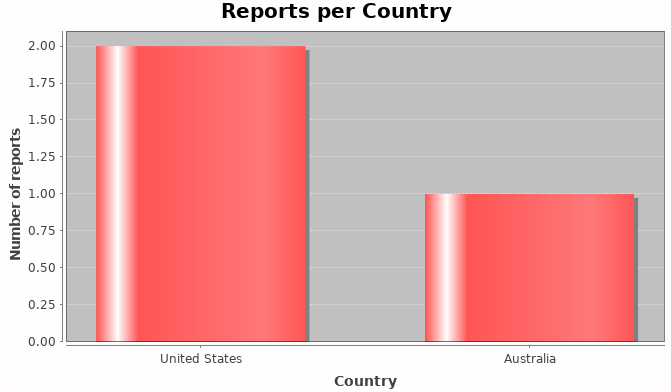


Figure 64. Reports per Country Bar Chart

A very useful statistic that might help find correlations between hate speech and specific countries. This can help us understand how different education systems from different countries may affect hate speech incidents. This will also help us better tackle the issue, by being able to tackle the education system of each country independently.

Unfortunately, only 3 out of the 44 reports that we used contained location data and therefore the sample is very small to be able to make any correlations.

**Reports per Time Period**

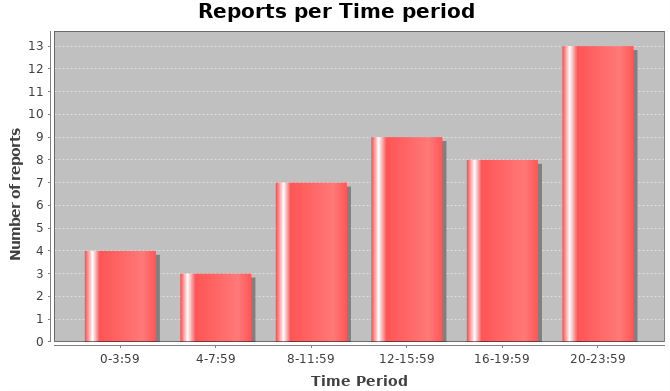


Figure 65. Reports per Time Period Bar Chart

For this statistic, the 24-hour day was divided into 6 different time periods. The time used for each report is the local time of each twitter user when he posted the tweet. The number of tweets per this time period was counted and presented.

As we can see from the above bar chart, the time 20:00 to 23:59 is the most active period regarding the report of tweets regarding hate speech, while the launch period 12:00-15:59 also seems to be very active.

By analyzing the time period, authorities or social media companies searching for hate speech content, can better target their searches, utilizing more resources in searching for hate speech during the most active time periods.

**Number of Reports for each Category per Time Period**

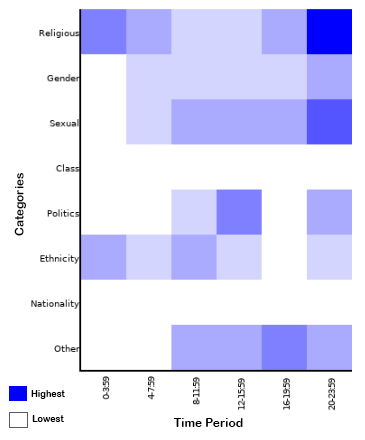


Figure 66. Reports per Category & Time Period Heatmap

Each row of the above table shows the most active time periods for each category. The darker the color, the greater number of reports for that specific time period.

For example, we can see that for the first row, the Religious category, the most active time period is between 20:00-23:59, since the darkest color appear in that column. While the most active time period for the Religious and Sexual categories seems to be the overall most active time period (20:00-23:59), other categories have different active time periods. The most active time period for the Politics category, for example, is the launch time period, from 12-15:59.

Although our data set contains only a few parameters and it too small, to make such conclusions, future work may be able to prove such correlations, or make new ones. Such correlations will further help in the fight against hate speech, since authorities and companies searching for hate speech content will be able to target their search even better, by searching for specific categories of hate speech in specific time periods.

Moreover, if groups of people that can be distinguished by specific attributes are somehow found to be more active in the time period that hate speech tweets of a specific category are most active as well, then perhaps further correlations can be made between certain groups of people and certain hate speech categories.

# Chapter 8

**Conclusion**

8.1 Conclusion

8.2 Future Work

## Conclusion

In this document, details about the methodologies that were used for the design and implementation of a smartphone application were presented. The main concepts of the application were described to be the convenience of the user experience and that concept followed through to end of the implementation. The main literature studied and the tools used are also presented, while describing the specified requirements that the application would need to meet in order to be successful, using natural language as well as use cases.

Following the requirements specification, the design and the architecture of the application were described, while screen Mock-ups from the application prototype were presented. In the development phase of this thesis, the development and implementation of the database and the different modules of the application were explained and the final developed interface was presented. Additionally, the database of the REST API was also described and the communication between the API and the application was explained.

A very important part of the application’s development life cycle is its evaluation. The whole reason behind the need for the development of this application was to provide something new to the users, a way of reporting that would be anonymous and convenient, to differentiate from other applications targeted towards hate speech reporting. The evaluation techniques used, proved that although the application can always be further improved, it does a very fine job of providing a convenient user experience.

Lastly, some very interesting statistics were presented, regarding reports of tweets that contain hate speech and although our data were discussed to be too small to make any convincing conclusions, they laid a path for further research and statistical analysis on this topic.

## Future Work

Regarding future work, we strongly believe that even better results could be achieved in the long run regarding the convenience of the application.

This application was created to be the start of what could later be a multipurpose platform covering not just the reporting process of hate speech content, but many more.

Since no approach to relevant hate speech authorities was made by the end of the implementation of this application, the different authority options for the user are written as Authority 1, Authority2 and Authority 3. Careful implementation took place, so that when an approach to relevant authorities is made, the sending of the reports to these authorities could be very easily implemented. Moreover, this careful implementation makes the sending of a report to the REST API to be a very easy process, allowing those authorities to easily receive the reports of the application, without having to write pieces of complex code. The authorities could even take the REST API we created and use it for themselves after making adjustments and connecting it their database.

In this digitized era, social media applications are being created faster than ever before. In the future, the application may also be extended to support a bigger variety of social media applications and platforms, to be able to attack online hate speech in all places and forms.

Lastly, an iOS application could be developed to provide iOS users the ability to conveniently report online hate speech. Although our application was decided to be an android application, due to the fact that the iOS operating system wouldn’t support many features that make this application as convenient as it is, we are certain that perhaps a similarly convenient iOS application could be created.

# Bibliography

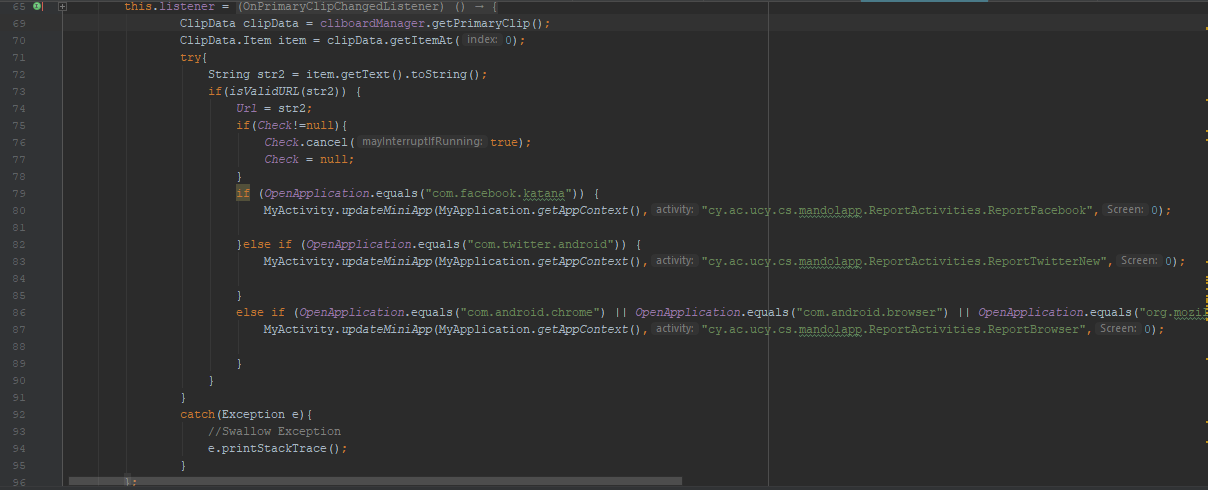
|  |  |
| --- | --- |
| [1] | M. G. CEO, "Number of worldwide social network users 2010-2021 | Statistic," July 2017. [Online]. Available: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/. [Accessed April 2018]. |
| [2] | O. dictionaries, "Definition of hate speech in English," [Online]. Available: https://en.oxforddictionaries.com/definition/hate\_speech. [Accessed May 2017]. |
| [3] | *Recommendation 97(20) of 30 October 1997.* |
| [4] | C. Callanan, " FAQ on Responding to on-line hate speech," 29 July 2016. [Online]. Available: http://mandola-project.eu/m/filer\_public/1a/af/1aaf50d3-8a38-40f4-b299-9c343f16cea1/mandola-d41.pdf. [Accessed May 2018]. |
| [5] | D. G. D.Stefanides, "MANDOLA Monitoring and Detecting Online Hate Speech Smartphone app," May 2017. [Online]. Available: http://mandola-project.eu/m/filer\_public/9e/09/9e09f655-1c23-4da6-928a-ec534a946867/mandola-d339.pdf. [Accessed April 2018]. |
| [6] | G. Sterling, "Nearly 80 percent of socia media time now is spent on mobile devices - Marketing Land," 4 April 2016. [Online]. Available: https://marketingland.com/facebook-usage-accounts-1-5-minutes-spent-mobile-171561. [Accessed April 2018]. |
| [7] | E. D. Marco, "Definition of illegal hatred and implications," 29 September 2017. [Online]. Available: http://mandola-project.eu/m/filer\_public/8d/b3/8db3c018-4e9f-4120-95cc-bf80b38f1749/mandola\_dd21b.pdf. [Accessed May 2018]. |
| [8] | "ABOUT eMORE," [Online]. Available: https://www.emoreproject.eu/about-project/. [Accessed May 2018]. |
| [9] | "About C.O.N.T.A.C.T and hate crime – Reporting Hate," [Online]. Available: http://reportinghate.eu/en/about-us/. [Accessed May 2018]. |
| [10] | T. I. N. A. CyberHate, "What We Do," [Online]. Available: http://www.inach.net/what-we-do.html. [Accessed May 2019]. |
| [11] | N. H. S. Movement, "About The Campaign," [Online]. Available: https://www.nohatespeechmovement.org/. [Accessed May 2018]. |
| [12] | T. N. K. C. Posted by Xavier Ducrohet, "Android Studio: An IDE built for Android," 15 May 2013. [Online]. Available: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html. [Accessed May 2018]. |
| [13] | "Tooleap Developer Center," [Online]. Available: https://developer.tooleap.com/. [Accessed May 2018]. |
| [14] | Twitter, Inc, "Build mobile apps with Twitter Kit," [Online]. Available: https://developer.twitter.com/en/docs/publisher-tools/twitterkit/overview. [Accessed May 2018]. |
| [15] | "Proto.io - Prototypes that feel real," Protoio, Inc, [Online]. Available: https://proto.io/. [Accessed May 2018]. |
| [16] | b. P. Github, "PhilJay/MPAndroidChart," [Online]. Available: https://github.com/PhilJay/MPAndroidChart/wiki. [Accessed May 2018]. |
| [17] | O. Shelajev, "Sparkjava is an amazing Java web framework. Do you really need it?," 23 March 2016. [Online]. Available: https://zeroturnaround.com/rebellabs/sparkjava-is-an-amazing-java-web-framework-do-you-really-need-it/. [Accessed May 2018]. |
| [18] | "Distribution dashboard | Android Developers," 8 May 2018. [Online]. Available: https://developer.android.com/about/dashboards/. [Accessed May 2018]. |
| [19] | S. Sebastian, "Devices without "Apps using Usage Data" or android.settings.USAGE\_ACCESS\_SETTINGS intent," 21 August 2015. [Online]. Available: https://stackoverflow.com/questions/32135903/devices-without-apps-using-usage-data-or-android-settings-usage-access-setting. [Accessed May 2018]. |
| [20] | "Measure App Performance with Android Profiler | Android Developers," 16 April 2018. [Online]. Available: https://developer.android.com/studio/profile/android-profiler. [Accessed May 2018]. |
| [21] | A. Muscara, "What to Ask Your Beta Testers to Get Better Feedback – Instabug Blog," 21 May 2017. [Online]. Available: https://blog.instabug.com/2017/05/what-to-ask-mobile-app-beta-testers/. [Accessed MAy 2018]. |
| [22] | WhatIs.com, "What is Amazon EC2 instance? - Definition from WhatIs.com," April 2014. [Online]. Available: https://searchaws.techtarget.com/definition/Amazon-EC2-instances. [Accessed May 2018]. |
| [24] | D. P. G. D. Stefanidis, "MANDOLA Monitoring and Detecting Online Hate Speech. Smartphone app," 2017. |
| [25] | C. STAMFORD, "Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013," 14 May 2013. [Online]. Available: https://www.gartner.com/newsroom/id/2482816. [Accessed April 2018]. |
| [26] | M. G. CEO, "Number of Google Play Store apps 2017 | Statistic," December 2017. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed April 2017]. |
| [27] | "Platform Architecture | Android Developers," 25 April 2018. [Online]. Available: https://developer.android.com/guide/platform/. [Accessed May 2017]. |

# Appendix

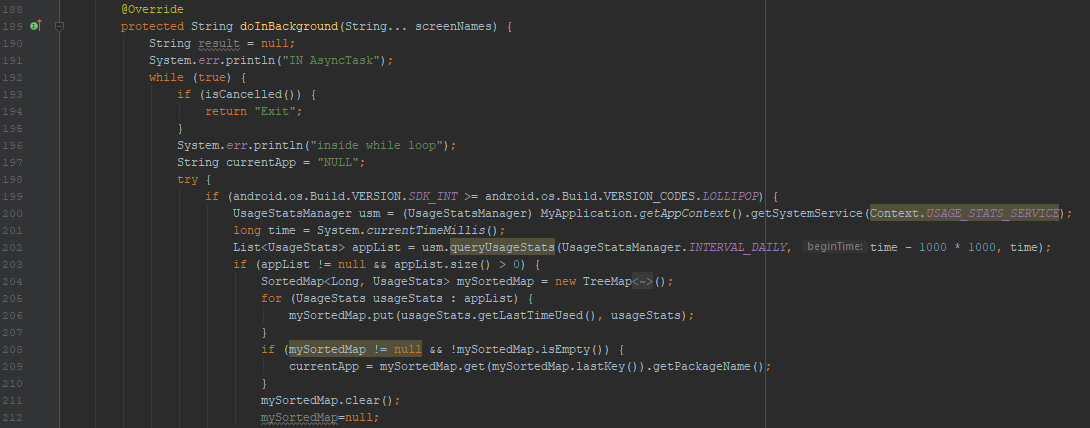
## Appendix A

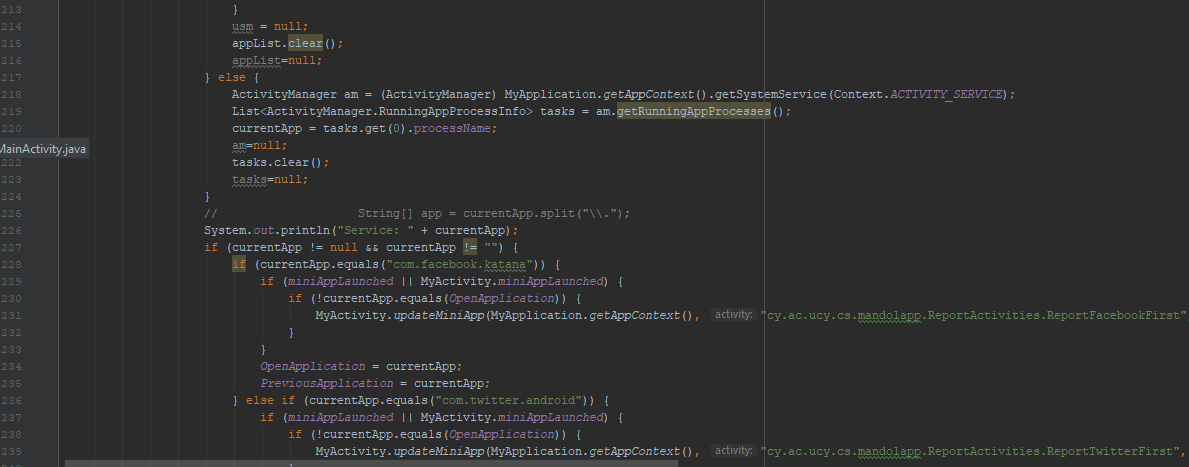
In the appendix some source code the system is going to be presented.

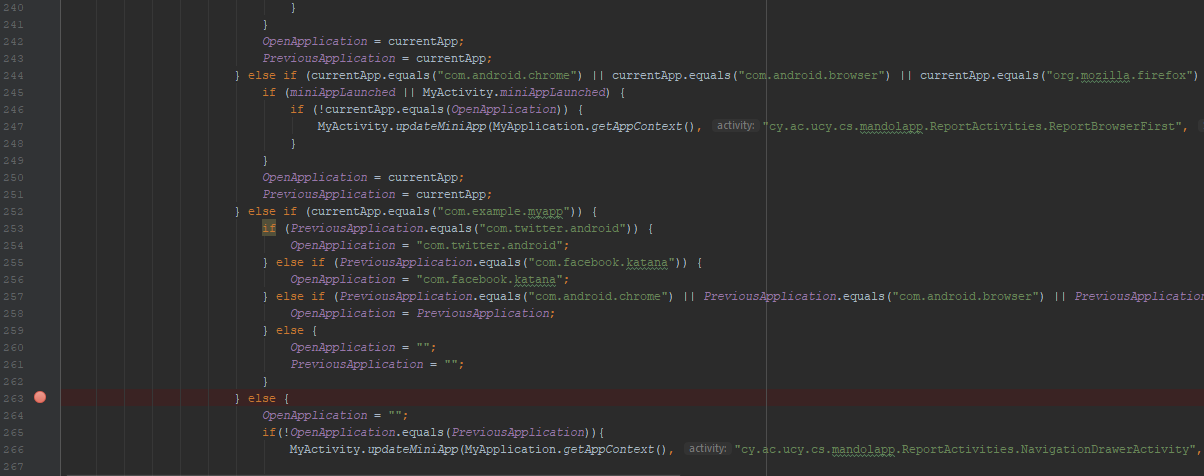
**Background Service - Listen to Copy events**



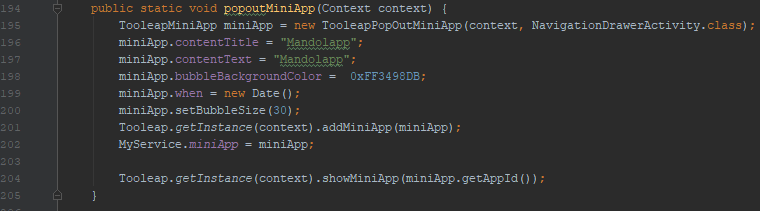
**Background Service – Get Foreground Application**



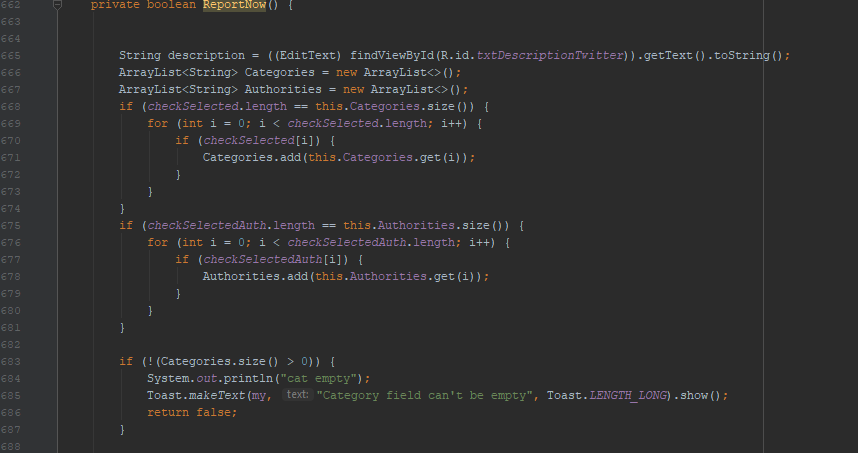


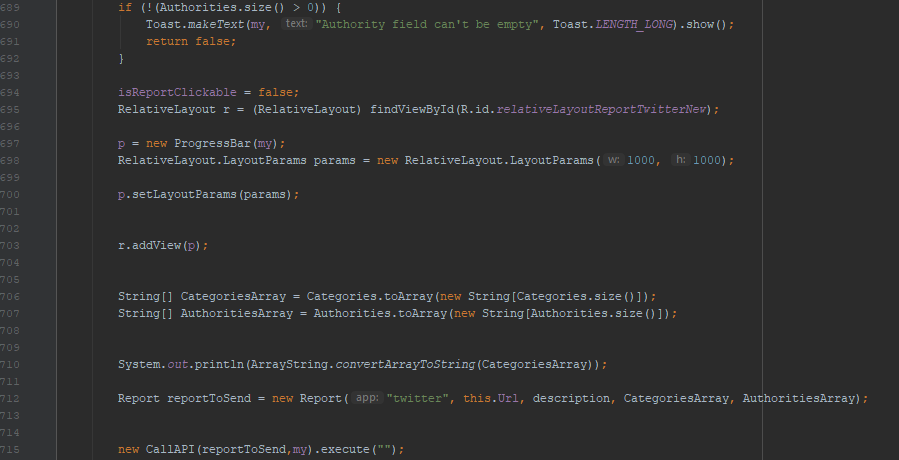
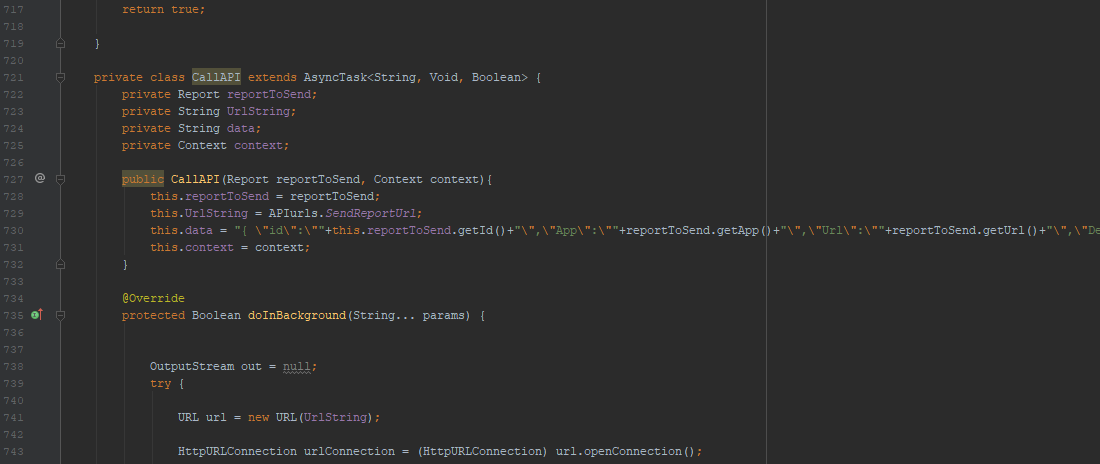


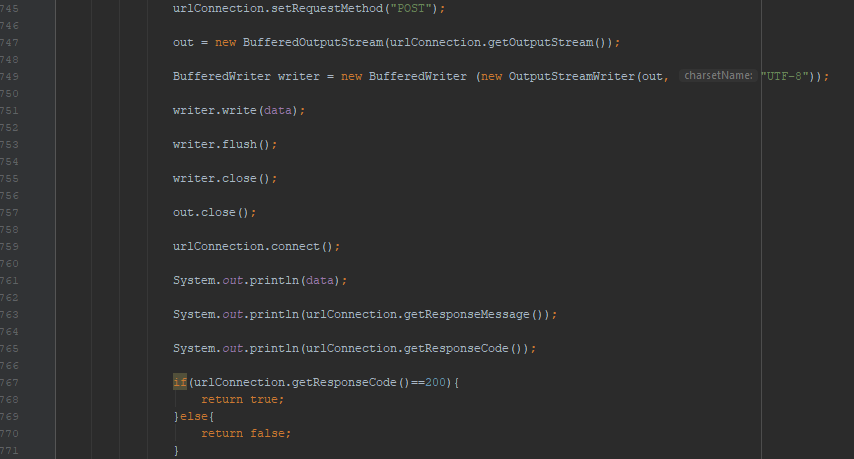
**Launch Mini App**

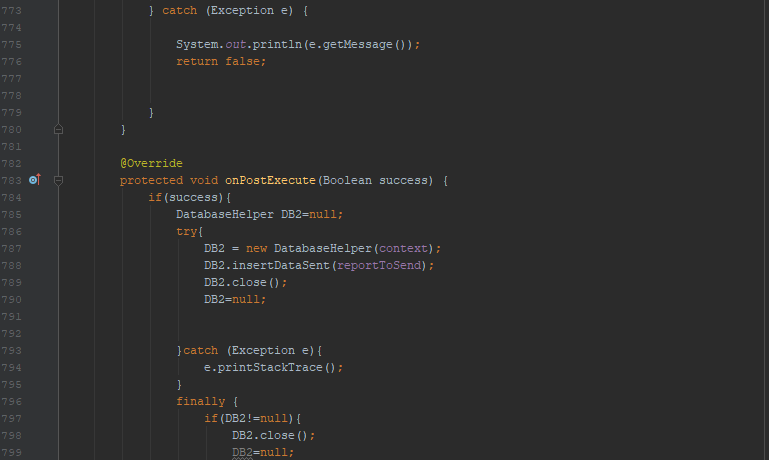


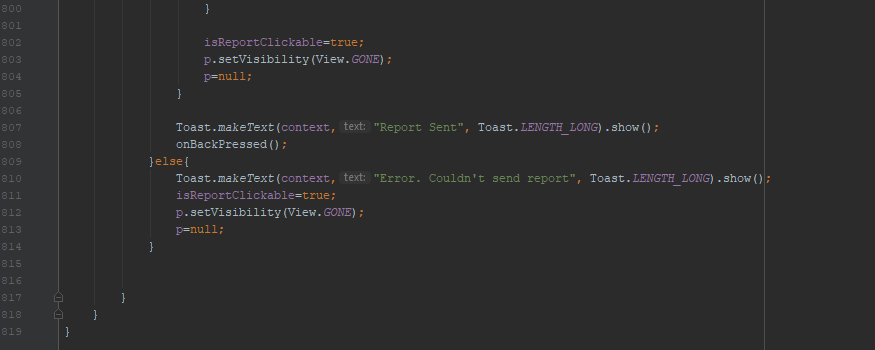
**Send Twitter Report to the API**









## Appendix B

In this appendix the Questionnaire given to the beta testing participants is presented

