School of Electronic Engineering and Computer Science

MSc in Software Engineering
Project Report 2015

News Anchor: News Aggregation and Content Curation

Rafael Alves Sa

August 2015

Abstract

The purpose of this report is to document the whole process of development of the MSc Project, from the initial research to the final stages of development. This project covers the subject areas of news aggregation and content curation, and consists of a web application developed in Dart and using Polymer web components.

The initial steps consisted of gathering user requirements and analysing some of the news aggregators available online to understand their limitations and good practices. The requirements were then specified to provide the core components of the application and introduce new features and concepts, which were developed iteratively following a spiral model of development.

There are a few limitations in comparison to currently existent applications, but News Anchor provides a centralised product that introduces a few new concepts which are not available on any other service such as story reactions and a blacklist. In addition to that, News Anchor relies on Twitter and Google News to provide not only professionally-written stories, but also stories that are normally published on Twitter before anywhere else, so that the user can easily stay on top of the news.

Keywords: News aggregation, content curation, social news, Twitter, Google News, Dart

Table of Contents

1. Introduction

1.1. Motivation

News aggregators have existed as computer software and on the web for over a decade to facilitate the access to news content. Instead of visiting each news publisher's website to find out whether there are any updates, news aggregators allow the content to be collected and delivered as it is published. Combined with automated and manual processes of content curation, the news can be tailored by selecting the most relevant stories from a specific source or related to a specific topic before they are delivered to the end user (Isbell, K, 2010).

Some of these existing services include NewsNow, Google News, Newsmap and News360, as well as RSS feed readers such as NewsBlur, Feedly, Digg Reader and NetNewsWire (Section 3). The main functionality of these news aggregators is the display of story headlines which link to the original source of the content, and some of them also embed the stories with personalised styles and views in order to adapt the content to the application. They may also provide other features to allow users to personalise and specify the content they want to see, use geolocation to arrange personalised versions of the service to specific countries and locations, sharing content with other people, adaptation of the content to different devices, saving stories to read later, and many others.

While the main value of these services is the aggregation and curation of varied content from multiple sources, some of them lack some features and good practices which can be found in similar services. They can also be restricted to specific devices, and the great majority of them do not seem to offer enough social capabilities apart from sharing stories on social networks. This is where News Anchor hopes to add more value to the existing market, not by reinventing the algorithms behind the aggregators already available, but by combining the power of some of them and making it available on any device with a modern web browser. On top of that, users are able to filter stories by the topics they like, search for local news or stories about any topic, and also take advantage of new features that will be introduced in the following sections of this report.

1.2. Objectives

This project consists of the development of a web application that allows users to follow the news in a single application and on any device. In order to deliver that, the system is divided into core components and extensions.

The core components are responsible for retrieving content and making it available to the client. They consist of a proxy server and a crawler. The proxy server handles requests made by the web client, while the crawler retrieves new stories from Twitter and Google News. In addition to this, there are

administrative tools that allow the system administrator to manage the contents of the database and add new topics and Twitter users.

The web application delivers a simple and easy to use experience where the user is able to get short headlines of stories, filter them by topic, add topics to the blacklist and access public reactions to stories. More details about the requirements are included in Section 4.2.

In terms of research, it aims to understand what the limitations and best practices are by analysing related services, as well as learn more about what could be done to improve the current services and user experiences by studying the user requirements.

1.3. Overview of Structure of the Report

The remainder of this report is organised as follows.

An introduction to the project's main subjects and their related concepts is given in Section 2 to help the reader understand the contents of this report. A literature survey is included in Section 3, where several commercial services are thoroughly analysed and critically compared against News Anchor. Section 4 is focused on the method used for the development of this project, specifies all the system, functional and non-functional requirements of the core modules and extensions of the application, and gives a short overview of

the framework design. A description of the implementation and its design, and a discussion of the methods used, issues encountered and testing of the core components and system extensions is provided in Sections 5 and 7. Finally, Section 7 gives an overview of the goals achieved, work that could be done in the future, and final conclusions.

2. Background

Aggregation is the process of collecting information from multiple sources and of different types, which in the context of this project means the collection of news articles about various topics, published by different publishers, and distributed in several formats.

Content curation is the process of collecting, organising and making sense of information.

An Application Program Interface (API) is a collection of functions, tools and resources that can be used to develop software, while also allowing code to be reused and more easily maintained.

The JavaScript Object Notation (JSON) is a format broadly used to transmit data over the Internet. It is a notation that easily allows object to be converted into a textual representation.

The Really Simple Syndication (RSS) is generally used by websites that frequently contain new updated information, such as newspapers and blogs, to make it easier to keep track of the updates.

These and other concepts will be explained in more details throughout this report.

3. Literature Survey

3.1. Google News

Traditional news readers select the news providers they know and like first, and only after they select the news articles they would like to read. Google News takes a different approach by displaying the headlines sorted by subjects and categories, and below each headline there are links to the respective articles published by different news publishers. This way the user can focus on the news stories rather than on their sources, and is still able to choose the news publishers that they prefer.

Google News allows different types of personalisation. The user can browse and add their favourite topics and publishers, and choose the amount of news about a specific topic or from a particular publisher to be displayed, as well as sort them the way they want. It also learns what the user's interests are based on what they keep reading, so that over time more stories from the user's favourite topics are shown. Besides, the customisations and all the settings are stored online and can be accessed from any device or browser after the user has signed in.

Where available, and if the user has allowed, Google News will use the user's location to present a local edition of the service with a local news section and

relevant stories for that location. The location can be detected automatically or selected manually by the user.

For breaking news and top stories, Google News also offers real-time coverage where the user can find more in-depth articles, more sources, citations and timelines of articles with a graph showing the distribution of the publication of related articles over the past few days. All this information is updated automatically and continuously as new articles are being published. However, the user can pause the automatic updates at any time.

Google News also comprises weather forecast updates, alerts – which can be sent to the user's email address or delivered as an RSS feed –, notifications on mobile devices using the app, and the ability to share stories on popular social networks, subscribe to RSS or Atom feeds, and search from historical news archives.

In terms of restrictions, it does not allow the user to bookmark articles to read later, add external sources from, for instance, RSS feeds, or pay for magazines and newspapers subscriptions. It also does not provide offline access, and the only way to share stories and interact with other users is by using external social networks. Moreover, Google News only acts as an aggregator of headlines with short previews. In order to read the full article, the user is redirected to the original source's website. On a mobile device, this means opening the article in another application – the browser.

Overall, the main strengths of Google News are its algorithm for aggregating, collecting and sorting news, including related and similar news from different sources and publishers, automatic detection of the user's location and the ability to configure notifications. Its main weaknesses are the lack of personalisation, social features and interactions with the stories, such as saving stories to read later. The user interface is also too basic and has an unappealing look.

3.2. Yahoo! News

Similar to Google News, Yahoo! News displays headlines of news articles, which are organised by categories and also link to the publishers' websites. However, the content personalisation works in quite a different way and is much more limited. On top of each headline there are three buttons: one is used to save a story, and the other two allow the user to request more or fewer stories of a topic – it asks the user for the category they would like to read more or less about. However, the choice is limited to just a few categories available.

The headlines are normally accompanied by images, videos or slide shows, and there are several sections dedicated exclusively to videos and other types of media. The user preferences are also synchronised across the several devices where the user is signed in, and it also provides weather forecast

updates and news notifications, although the latter are only available on the application for mobile devices.

In spite of the similarities with other news aggregators, Yahoo! News has many more limitations, is less customisable and provides fewer features. It appears that the ability to save stories is the only feature which is not available on Google News.

Overall, its main strengths are the ability to like or discard individual stories in order to improve the accuracy of the topics that the user is interested in, and the way stories are organised by media type. Its main weaknesses are the limited selection of topics available, the lack of features on the mobile application which are present on the desktop version, and a user interface that could be more appealing to the user.

3.3. News360

The home page of News360 displays stories which the user might be interested in from various different categories. Here the articles are not organised by category; instead they all stand next to each other forming a wall of images with headlines, tags and short previews. There is a menu where the user can select a specific category and read stories of that category only.

For each article the user can like it, dislike it, save it and share it. The like and dislike buttons are used to improve the recommendations, so that more articles that meet the user's interests are displayed. Articles can be saved to be read later, and they can also be shared on popular social networks and via email.

Similar to other services, News360 uses the user's location to provide local editions of the website, when available, with local news. Although there are only a few countries available for the local editions, there is an international edition. The location can be set manually by the user.

Apart from categories, the user can also add their favourite publishers, but this is not as straightforward as adding new categories. This is because a source needs to be added via an article that has been published by it, or by using the general search box to manually search for it – which will also include other content, not just publishers. Unlike categories, publishers are not featured or suggested on the Explore page, and they are displayed alongside all the topics without any distinction.

The most interesting aspect about News360 is that all articles can be fully read without being redirected to the publisher's website. Instead, a new window is open with a News360 toolbar on top that includes all the functions previously available plus a list of related articles, and below the original publisher's website is loaded with the selected article. Thus, the publisher can

still make profit out of its own adverts, and users will still be able to use News360 features.

News360 also has an application available for mobile devices. It has a very similar interface to the web version, and includes basically the same features. In addition, when an article is saved on the mobile app, the user is still able to read it even when offline. However, neither the desktop nor the mobile versions provide notifications or weather forecasts.

Overall, the main strengths of News360 are the ability to load stories inside a frame without leaving the website so that the user can still take advantage of News360 features such as finding similar stories. However, the user experience can be quite confusing, especially when adding new publishers and selecting content sources.

3.4. NewsBlur

NewsBlur is an RSS feeds reader, and therefore the user needs to add their own sources in the first place, be it blogs or online newspapers. A new source can be added by typing a title or URL, and several suggestions start appearing as the user is typing. If a source cannot be found just by typing a name, the full URL must be provided.

The sources can be organised into different folders and subfolders, and the content can be sorted by the number of subscribers, frequency, recency and number of times opened. The sources are then displayed on a left sidebar accompanied by the number of unread stories, and the user can filter them in order to display all, only unread and not yet rated, liked, or saved stories. Globally shared stories and stories suggested by friends also appear on the sidebar.

There are four modes to read stories: original, feed, text and story. The original mode shows the website's home page without focusing on any story in particular. The feed mode displays the story as it would appear on the original website but without any formatting, although still including images, videos, etc. If the full story is not provided by the source, a link at the end of the preview will take the user to the original web page of the story. The text mode is a premium feature that only displays the plain text of the story without any adverts or distractions. Finally, the story mode shows the original web page of an individual story.

NewsBlur allows the training of content, i.e. the user can like or dislike several characteristics about a story such as its title, tags, authors, publishers and the person who shared it. The more the user trains NewsBlur, the more articles that match their interests will be displayed. So, for example, if the user is following a blog that covers more than one topic, they can expect to see more

stories about the topics that they like and fewer stories about the topics which they are not interested in.

Additionally, stories can be saved and also shared, although the latter works in a different way compared to other news services: instead of sharing news on social networks, the user is prompted to write a comment, which will then be displayed underneath the story. Moreover, the user's profile will include the stories they have shared as well as their comments, and this can be shared with anyone, including people who do not use NewsBlur.

Overall, the best qualities about NewsBlur are its many features and customisations, the complete control over the sources, and the built-in social interactions such as a user profile and comments on stories, all available either on the web or on the mobile app. However, it does not provide content based on location, offline reading or content suggestions on any platform.

There are other services such as Feedly and The Old Reader, based on the discontinued Google Reader, which I am not going to review due to the many similarities with NewsBlur.

3.5. Google Play Newsstand

Google Play Newsstand is only available as a mobile application. It accommodates both newspapers and magazines, and provides free content

as well as paid subscriptions. The content and their sources are organised by categories, which makes it easier to explore and add new sources, and also to read articles from a specific topic when the user is only interested in that topic.

When a headline is selected, the full story is displayed as it was published on the original website, including images, videos, audio and adverts, although presented in an optimised view. Optionally, it can be opened on the browser and read on the publisher's website. Similar to other services, the application learns the user's interests based on their reading habits, and related content which might interest the user is suggested at the end of each article.

Articles can be shared via multiple other applications installed on the user's device, bookmarked to be read later, and even translated into a different language. Offline access is also available for any news edition, topic or magazine, and the user can receive notifications when new content is available.

In summary, the main strengths of Google Play Newsstand are the selection of free content as well as paid subscriptions, the ability to open stories inside the application without having to load them on the browser, and being able to share stores on several other applications, translate them into other languages and save them to read them later even offline. The main weakness is that the application is only available on mobile devices.

A very similar application is Apple News, a new mobile application recently announced which is going to be available on Apple iOS 9.

3.6. Digg

Digg has a very minimalistic interface and concept. As opposed to other services, the stories are picked up and sorted automatically for the user. It fetches the most interesting and discussed stories on the Internet, as advertised on their website, from various proprietary data sources, and delivers them to the web, to the mobile application or via a daily email.

Registered user can ‘digg’ a story, i.e. give positive feedback and increase its visibility. This builds up its Digg score, which combined with the number of shares on social networks and other factors makes a story more or less popular, and this determines where it appears on the website and mobile app: the higher a story’s Digg score is, the more visible it becomes. Therefore, the stories that end up being displayed on Digg are automatically selected by algorithms and networks as well as by people’s reactions to stories all over the Internet.

In addition to ‘digging’ a story and sharing it on social networks, the user can also save it to read it later, and organise stories and sources into folders.

There are no additional features or social interactions available, and in order to read a story, Digg redirects the user to its original web page.

In summary, Digg's strengths are its simplicity of use and minimalistic user interface, as well as the quality of the content that is selected by its algorithms and user ratings. Its weaknesses are the lack of basic features and personal customisations such as filtering stories by topic or location.

3.7. Conclusions

After individual analysis of several news aggregators and applications, it seems that the good practices include algorithms which are capable of finding related stories from multiple sources and publishers and removing duplicates, filtering stories in order to promote the ones that match the user's interests, using location to provide a localised version of the service with more relevant stories, and the ability to personalise the topics and sources of the stories based on the user's preferences.

In terms of the way the news is delivered, the application should be simple, easy to use and appealing to the user, and a short, descriptive headline summarising what the story is about should be provided when the application redirects the user to the website where the story was originally published. In contrast, some of the applications such as NewsBlur (Section 3.4) support the display of the full contents of the story inside the application, though this is not

always possible without a paid subscription due to terms and conditions of use, copyright issues and the publishers granting permission.

News Anchor relies on Twitter and Google News to select and retrieve the stories that are delivered to the user. In comparison with the analysis of applications in the sections above, News Anchor is not as powerful in terms of tailoring news to match the user's interests and offers fewer personal preferences for content personalisation, for the topics and Twitter users available are preselected by the system administrator. This is to reduce the amount of irrelevant content which can be found on Twitter in particular, even though the user is able to search for any topic without restrictions. Nonetheless, the system administration can add any topic or Twitter user to the database so that their stories become available to users.

In terms of supported media types, Yahoo! News (Section 3.2) seems to support a broader range of media types than any other of the services analysed. Unlike most of the other applications—News Anchor included—which only support text and images, Yahoo! News also aggregates videos and slide shows related to news, and includes sections dedicated to those specified media types.

News Anchor also takes advantage of location to display stories relevant to a particular location. However, rather than detecting the user's current location as happens on Google News (Section 3.1) and News360 (Section 3.3), News

Anchor requires the user to enter a location. This allows them to quickly find stories from any place in the world, including the user's current location.

Despite these limitations (see Section 7.3), the analysed services are unable to solve some of the problems that News Anchor aims to solve (see Section 7.2). Since it relies on Twitter and Google News, users can have access not only to stories written by professional, well-known publishers, but also to social news, i.e. stories published on Twitter which would normally be discarded. The greatest advantage of using this concept, which in this case is supported by Twitter, is that users can have access to news as they happen and much quicker than they would normally be published on other sources.

News Anchor also introduces two new features which are not currently offered by any other service available on the market: a blacklist and story reactions. The blacklist allows topics to be hidden from the story wall, and story reactions provide a simple way to understand the impact the latest news is having on people. These will be thoroughly explained in Section 7.2. Moreover, all the features are available on the web, which means users can have access to the service and all its features on any of the device they use, including mobile devices.

4. Method

4.1 Development Methodology

The development methodology adopted for the development of this project is the spiral model (Boemh, 2000). It involves several iterations with identical stages—planning of the requirements to implement, identification and evaluation of possible approaches, development and testing, and planning of the next phase—, each of them introducing new requirements from the outlined project specifications.

The first phase consisted of developing an early prototype of the user interface and implementing a proxy server to enable the application to request resources from entities located outside the application domain. The way this process works is discussed in more detail in Section 5.3. With the Tweets now being retrieved from Twitter, it became possible to develop the rest of the user interface and test how the Tweets would be displayed on the application, including parsing and filtering the text, converting encoded text URLs into links and displaying photos when available.

The second phase focused on developing the first implementation of the database, capable of storing all the necessary, relevant data from Twitter: users' details such as name, screen name and location, and data related to Tweets, including its contents, date of publication, photos, URLs and hashtags. This was followed by administration tools on the server side that

allow the system administrator to add new Twitter users to the database, as well as all the core Twitter functions which would support the following phases of development. A few examples are the functions to parse the JSON contents in the Twitter response, retrieve the hashtags, URLs and photos, and store them in the database.

The third phase introduced the implementation of a Twitter crawler. It would crawl Twitter every five minutes, request new Tweets published after the last one retrieved, and if there were any new Tweets, it would store them in the database.

In the fourth phase, tables were added to the database to support Google News stories, and core functions were developed to interact with the database, allowing the storage of stories and the system administrator to add topics to the database and manage Google News data. After this, the crawler developed in the previous phase was updated to crawl Google News as well.

In the fifth phase the database was updated to include a higher level entity, *Story*, and the entities *Tweet* and *GoogleNewsStory* became its sub-entities. After testing this new implementation, the proxy server was updated to retrieve Tweets and Google News stories from the database instead of making individual HTTP requests to the original sources, and later convert them into a Story object in the JSON format, which could then be sent as a string to the web client.

The last phase focused on updating the web application on the client side to display the stories and add features such as filtering news by topic and location, search, user reactions and blacklist. In order to achieve this, the database had to be updated to include new tables for story reactions and the blacklist. It also introduced some changes to the layout to make it compatible with all mobile devices of different screen sizes.

4.2. Requirements

The system consists of a client and server applications. The server is responsible for handling requests made by clients, crawl Twitter and Google News for new updates, and store and retrieve data from the database. The client is a web application that is responsible for delivering stories to the user.

The system requirements of the client side are an Internet connection and a modern web server capable of interpreting HTML5, CSS3 and Dart or JavaScript applications, since the code written in Dart can be translated into JavaScript code. The recommended browsers are Internet Explorer (version 10 and above), Firefox (version 40 and above), Chrome (version 44 and above), and Safari (version 6 and above).

The system requirements of the server side are an Internet connection, the Dart virtual machine (VM) (version 1.11 and above) and a MySQL server (version 5.6 and above).

4.2.1. Core Requirements: Crawler and Proxy Server

The system core comprises the proxy server and administrative applications that communicate directly with the database, as well as the Twitter and Google News crawler. Only the system administrator should have access to these administrative tools, whose responsibility is to add more news sources to the database, and they shall not interfere with the normal functionality of the other components of the system.

The system shall have command-line functions to add new Twitter users to the database and display the existing ones. In terms of functions to support other server applications, it shall be able to request Tweets and details about users from Twitter. If this is successful, Twitter will send a response in JSON containing full user and Tweet details, including hashtags, URLs, and photos, and therefore the system shall provide functions to parse and filter the JSON contents that are relevant to the application domain, and convert them into equivalent Dart objects.

After this, the system must be able to store Tweets and Twitter users objects in the database. In addition to this, the system shall be able to verify whether a Tweet or a Twitter user already exists in the database, so as to prevent duplicated data from being entered in the database or, even worse, the system failing with an error. The system shall provide similar functions with a

similar behaviour to add hashtags, URLs and photos for each of the Tweets added and when applicable.

The system should also have functions to retrieve all the Tweets stored in the database in chronological order by date of publication, search for keywords on Twitter, retrieve all the Tweets published by a user before a specific Tweet (this will be used to load more Tweets when the user has already read the most recent ones), request the newest Tweets (this should request all the Tweets published by a user after the last Tweet already stored in the database, or the latest twenty Tweets if there are no Tweets by that user in the database yet), and finally functions to obtain the hashtags, URLs and photos for a specific Tweet.

In a similar way, the system shall have command-line functions to add Google News topics to the database and display the existing ones. It should then be able to retrieve stories about those topics and add them to the database. Google News provides access to stories in RSS feeds, and therefore the system must be able to parse and convert the contents that are relevant to the application domain into equivalent Dart object.

The system shall provide functions to retrieve the latest stories about the topics stores in the database, about a specific location or about any other keyword. There should also be functions to verify whether a story already exists in the database in order to avoid adding duplicates and prevent the

system from failing with an error. The proxy server must be able to add, delete and send to the client the topics in the blacklist stored in the database, as well as update stories reactions in real-time.

Once all these auxiliary functions are supported by the system, the system should provide a Twitter and Google News crawler to keep the database updated with the newest stories published on these two sources, as well as a proxy server to act as an intermediary between the client and the external servers.

The crawler and the proxy server are going to be the core applications of the system, constituting the engine that powers all the operations supported on the client side, so it is of the most extreme importance that they are reliable, efficient, secure and available for as much time as possible.

The crawler shall check for Twitter and Google News updates every five minutes, and it should implement a fail-safe mechanism that prevents it from failing with errors in case there is an error retrieving updates. Moreover, if it is unable to retrieve updates from Twitter, it should still be able to retrieve updates from Google News, and vice versa.

The proxy server shall act as a gateway between the client and the entities outside the application domain. It should also implement a fail-safe mechanism to prevent it from failing in the event of an error, and these events

must be communicated to the client in the most appropriate manner to avoid it reaching a blocking state.

4.2.2. Extension Requirements: Web Application

The web application on the client side must be reliable, secure, available for as long as possible, and inform the user of failures in a meaningful, friendly way. The user interface should be simple and easy to use, and the user should not be able to notice any difference between a Tweet and a Google News story in terms of usability and interactions available.

The system should allow the user to filter stories by topic, browse stories from a specific location and search for any other topic besides the default ones. When geographical stories are selected, the system should prompt the user to enter a location to allow them to browse stories not only from their current location, but also from any other location in the world.

Stories should be displayed in chronological order by date of publication, and they shall display the publisher, a title for Google News stories, the date of publication, a short, descriptive headline or the contents of the Tweet, photos when available, and a link to the Twitter or publisher's web page with the original story. For each story, the system should allow the user to select whether their reaction to the story was positive or negative, as well as add it—and similar stories—to the blacklist.

Reactions is a social feature that will allow users to see what impact a story has had on other users and what the general opinion about a topic is. For instance, during the general elections for the Prime Minister of the United Kingdom, users would have been able to see what the other users' reactions to each of the parties' campaigns were, and what parties were among the favourite ones. The blacklist should allow the user to discard all the news about a specific topic. Going back to the example of the elections in the United Kingdom, if a user did not wish to read any more news about this topic they could simply add it to their blacklist, and at any point they could remove it from the blacklist to start reading about it again. To prevent the system from filtering out the wrong topics, it should ask the user to confirm the topic before it is added to the blacklist.

Finally, the system should be able to refresh the application with the latest news without the user having to refresh the entire web page.

4.3. Framework Design Overview

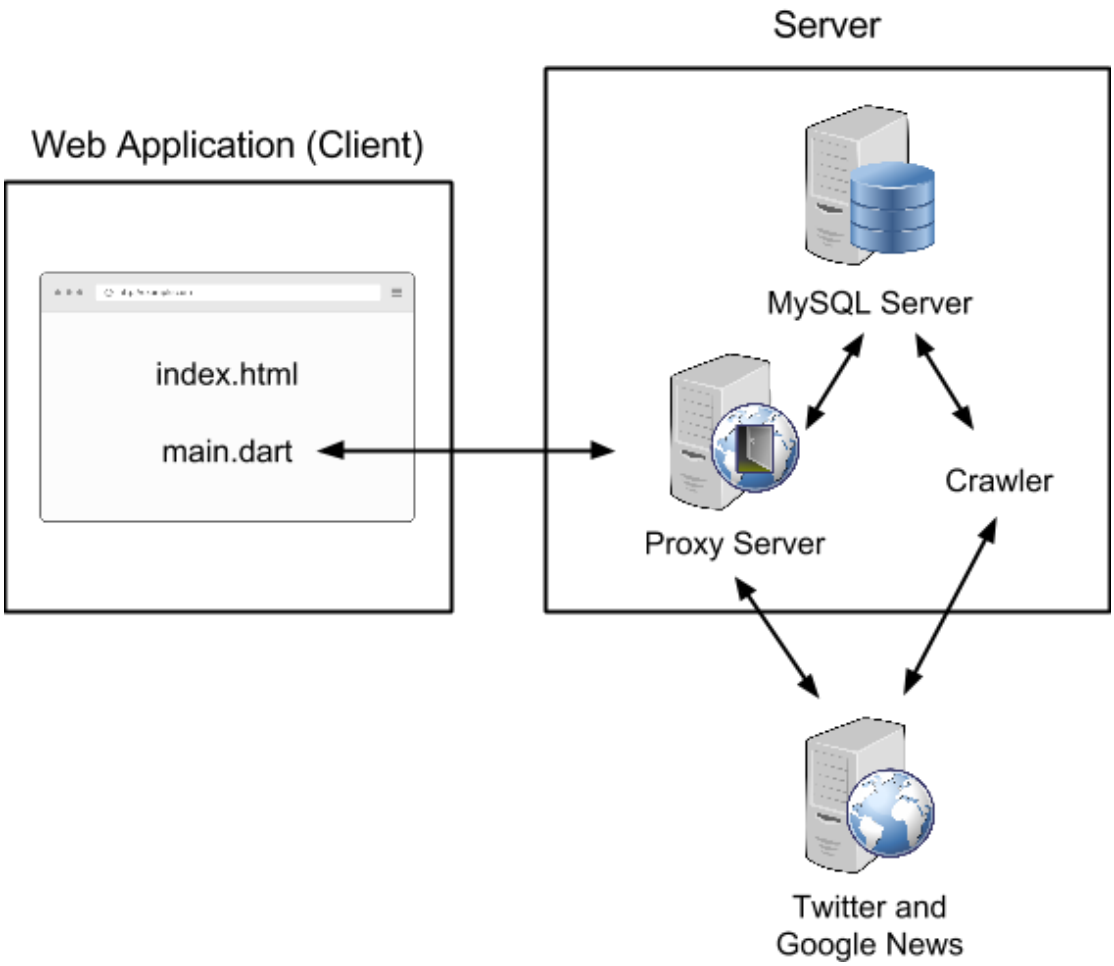


Figure 4.3.1. Model of the System Design

The core of the system comprises the crawler and the auxiliary functions to convert the data encoded in JSON and RSS feeds into native Dart objects and to communicate with the database. There is one main class: *Story*, and its subclasses are *Tweet* and *GoogleNewsStory*. These two subclasses implement constructors that allow parsing a JSON string into a Tweet object and an RSS feed into a Google News Story object, respectively.

As Figure 4.3.1 shows, the crawler retrieves stories from Twitter and Google News servers, and then after having parsed the data using the classes mentioned above, communicates with the MySQL database in order to store the stories. The proxy server normally retrieves stories from the database only, but it may also request data directly from Twitter and Google News when the user searches for a specific topic or requests stories from a geographical location.

The web application on the client side only communicates with the proxy server (see Figure 4.3.1). It simply makes requests to the proxy server, waits for its response with a JSON string, and then has to parse its contents and convert them into a Story HTML element before it is finally able to display the stories.

5. Core System: Crawler and Proxy Server

5.1. Design

There are three main classes: *Story*, *Twitter* and *GoogleNewsStory*, and the last two are subclasses of the class *Story*, which mainly defines the type of story: a Tweet or a Google News Story.

The class *Tweet* defines the properties ID (a unique identifier of a Tweet), user (an instance of the class *User*), date of publication, the text of the Tweet, URL, a list of hashtags, a map of URLs with the URLs included in the Tweet's text (where an internal Twitter URL is mapped to its display URL, i.e. the original URL as displayed in the Tweet's text), the media URL (the URL of the photo as included in the Tweet's text, and a list containing URLs of photos. Apart from *Tweet*, the class *User* defines the properties of a Twitter user account with a screen name (this is a unique identifier of a Twitter account, e.g. *@BBCBreaking*), name, URL of the profile image, geographical location and URL of the user profile (e.g. *https://twitter.com/BBCBreaking*).

The class *GoogleNewsStory* defines the properties *guid* (a unique ID that identifies the story on Google News), title of story, publisher, date of publication, the headline, the URL of the story's web page, and topic.

The proxy server and crawler applications were implemented separately so that they can both be executed in parallel at the same time. They rely heavily on the aforementioned classes and the APIs they provide. The crawler does not communicate with the database directly, but the proxy server does so when there are client requests to retrieve stories, manage the blacklist and update story reactions.

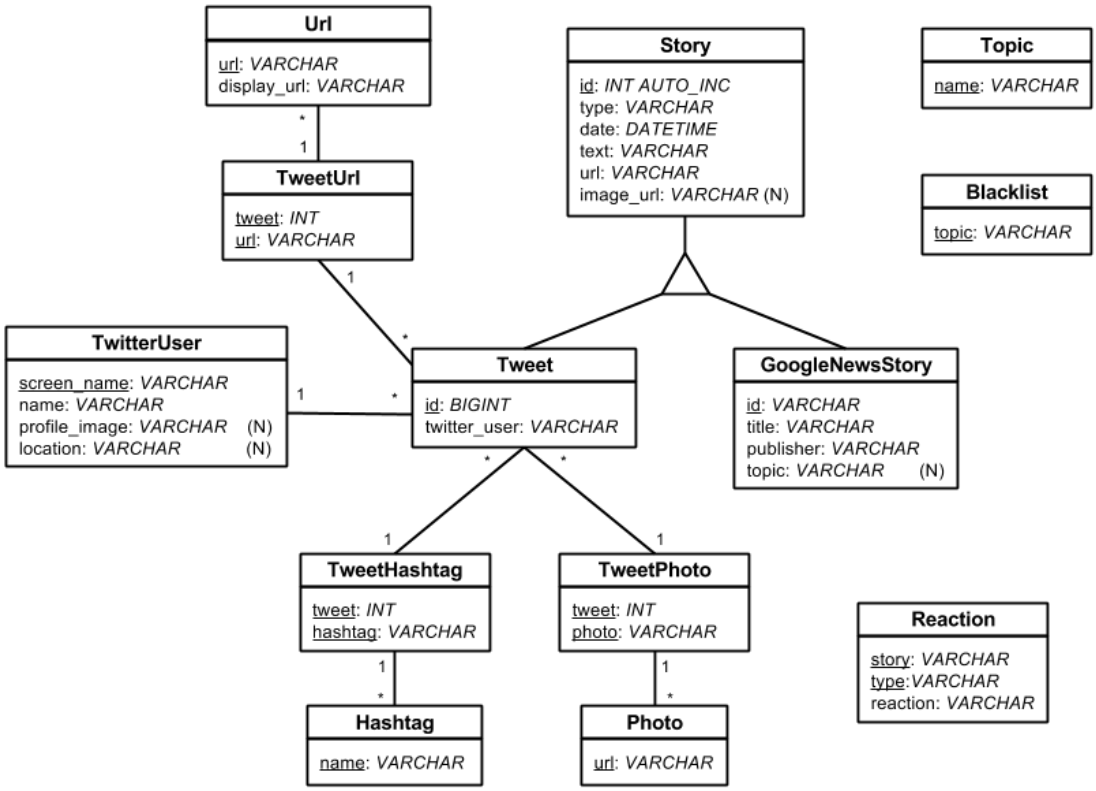


Figure 5.1.1. Database Model

The database is designed as shown in Figure 5.1.1. The underlined properties (e.g. '<u>id</u>') represent the entity's primary keys, the primary keys marked with *'AUTO_INC*' are automatically incremented for each new row inserted into the table, and the properties marked with '(N)' can be 'NULL'.

The entities *Tweet* and *GoogleNewsStory* extend *Story*, and therefore inherit its properties. The property ‘twitter_user’ of *Tweet* reference the primary key ‘screen_name’ of *TwitterUser*. Tweets can also have multiple hashtags, URLs and photos, and so the entities *TweetHashtag*, *TweetUrl* and *TweetPhoto* establish a relationship between a Tweet and it’s several hashtags, URLs and photos, respectively.

Apart from this, there are tables for story topics, positive and negative reactions to stories and a blacklist with topics that should be filtered out from the news wall.

5.2. Design Discussion

The crawler and the proxy server are powered by the APIs provided by *twitter_server.dart* and *google_news_server.dart*. They were kept in files separated from the crawler and proxy server implementations to reduce their complexity, as well as to permit its functions to be reused. This makes it easier to maintain the code, and also allows any new component later added to the system to access its public API.

The crawler was implemented as a separate application to allow it to keep being executed continuously without compromising or preventing other tasks from being executed by the server at the same time, such as the proxy server

handling requests from clients and communicating with the database, and the administrative tools which also communicate with the database.

The reason why the proxy server was implemented, as opposed to having the web application on the client side make the requests and communicate with the database directly, was due to the fact that requests of resources located outside the application domain are restricted by most web browsers to prevent security attacks and keep the users of the application safe.

However, it is still possible to request external resources by including a cross-origin resource sharing (CORS) value in the header of the HTTP response (see Section 5.3). Since this cannot be requested from Twitter and Google News directly, it is the proxy server's job to make the request on behalf of the client and add the CORS to the response header before forwarding it back to the client. Moreover, this implementation, which is often referred to as a thin-client architecture (Nieh, J. et al., 2000), frees the client from unnecessary responsibility, hence reducing it's complexity and making it easier to maintain and expand the whole system, while also making the overall system design more elegant.

Finally, in one of the last development phases, the database entities *Tweet* and *GoogleNewsStory* became sub-entities of *Story* in order to reduce complexity, and because on the client side it does not matter what the type of a story is, for they are all homogeneous and behave in a similar way. It also

made it easier to display all the stories in chronological order by date of publication.

5.3. Implementation

The proxy server is a gateway between clients and the Twitter and Google News servers, and it was implemented to continuously listen to HTTP requests made on port 8081. Upon receiving a request, it parses the URL parameters to determine whether the stories can be retrieved directly from the database, or whether new requests need to be made to Twitter or Google News, for instance to search for a topic or request news from a specific location. In that case, when the response is received, the JSON (in case of a Twitter request) or the RSS feed (in case of a Google News request) contents are converted into a native Dart object using the classes *Tweet* and *GoogleNewsStory*, respectively, so that the stories can be stored in the database.

When the resource is requested from Twitter, there is an extra step before the proxy server is able to make the request, which is shown in steps 2 and 3 in Image 5.3.1 (these do not apply to Google News requests). Twitter requires all requests to be authenticated using an OAuth access token (Twitter, 2015). This is requested by the proxy server the first time it is installed, and then the access token is stored in a local file so that it can be reused for all the

subsequent requests, which reduces the number of HTTP requests and increases the overall efficiency of the application.

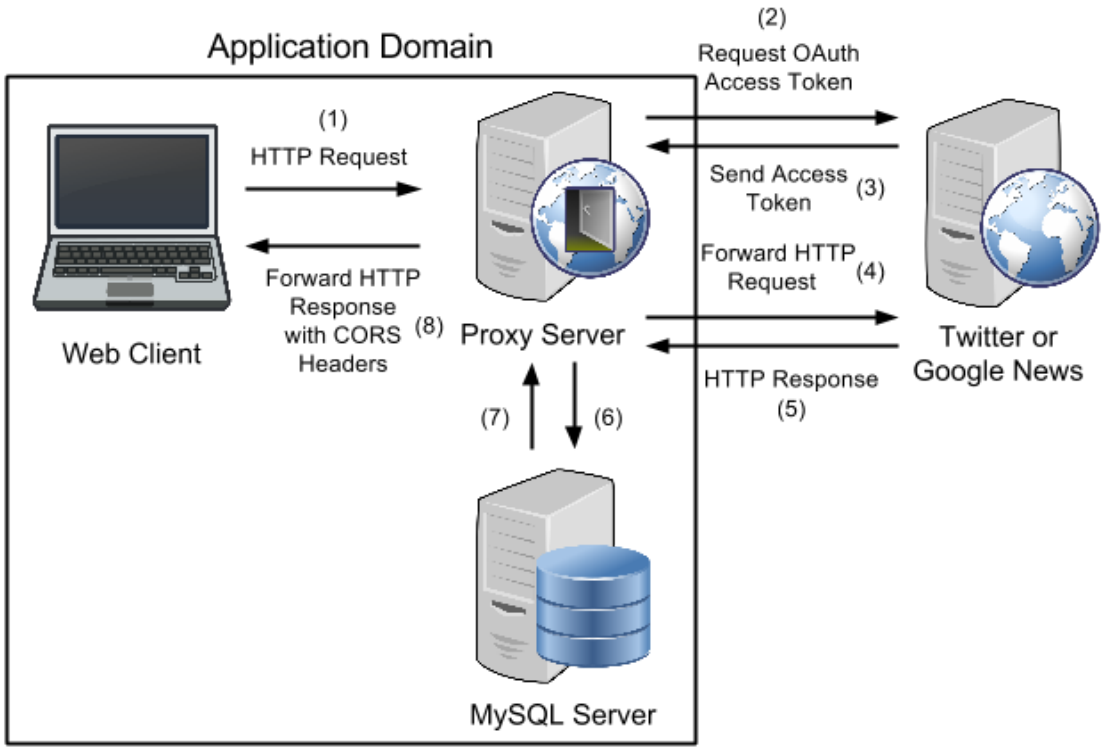


Figure 5.3.1. Interactions between the proxy server and other entities

Once the proxy server has retrieved all the stories, it needs to convert them into a JSON string that can be sent to the client over the network. And finally, the last thing it needs to do before the response is sent, is add the field CORS to the HTTP response header (Kesteren, 2014). This way the client is able to access the resources located from outside the application domain (see step 8 in Figure 5.3.1).

Apart from stories, the proxy server also handles requests related to story reactions and the blacklist. These two features are synchronised between the

server and the client, and updated in real-time. When the client makes a request and the blacklist is not empty, the proxy server omits from the results the stories that include the blacklisted keywords in the title and text of the story, Twitter hashtags and Google News topics. As for reactions, when a story receives a positive or negative reaction, it is automatically included in the Positive Reactions or Negative Reactions section, respectively. If the user updates or deselects a reaction, it is then included in the other section or deleted altogether.

The second core component of the system is the Twitter and Google News crawler. Is it implemented so that is looks for updates every five minutes. Just like the proxy server, the crawler relies on the functions defined in *twitter_server.dart* and *google_news_server.dart* to execute its tasks. It also implements a fail-safe mechanism to prevent it from failing when an error occurs. If an error occurs while retrieving Tweets or storing them in the database, the crawler is still able to do the same operations for Google News stories, and the opposite applies as well.

Because Twitter offers a REST API (Twitter, 2015), the crawler is able to request only the Tweets published by a specific user after the most recent one already stored in the database. This reduces the number of operations and improves the overall efficiency of the system. Oh the other hand, Google News only provides RSS feeds, and therefore the request queries are limited to topics, locations and a few other basic parameters. This means that the

crawler has to verify whether each story retrieved from Google News already exists in the database, and add the ones that do not exist yet. Either way, the functions that communicate with the database were implemented so that the system does not store duplicate stories or fail with an error.

In addition to the APIs provided by *twitter_server.dart* and *google_news_story.dart*, these files also implement the administrative tools that allow the system administrator to manage and add new topics and Twitter users to the database.

5.4. Discussion of Development Issues

During the initial development of the proxy server, sometimes the client was only receiving part of the contents sent by the server. After performing several tests, they revealed that the server was closing the HTTP connection before the whole content of the message had been written and sent to the client. This was caused by an incorrect use of Dart's asynchronous API.

Later on, and during the implementation of the administrative tools and functions to store and retrieve data from the database, the command-line operations were taking a few seconds to terminate the process after they had been completed. This delay that appeared to be a bug in the Dart VM was after all an issue caused by the MySQL connection pool which was not being closed at the end of the main method.

After the core components of the system had been developed and the stories were being loaded on the web application, some of the stories were displaying a broken image and the browser console was showing an error reporting invalid image URLs. This was caused by the Google News parser which was returning URLs for null values. The solution consisted of replacing null values of stories with no photos with a generic image.

Lastly, there were bugs retrieving stories and updating the reaction buttons after the blacklist and reaction features had been implemented. The MySQL queries were returning no results when there were no topics in the blacklist and/or no Twitter hashtags stored in the database, and the reaction buttons were not being updated at the same time as the updates were applied to the database.

5.5. Testing and Evaluation

All the functions were tested individually and after integration with the rest of the system as they were being implemented. One of the goals of the testing procedures was also to ensure that exceptions were being caught and handled appropriately in order to prevent the system from failing in the event of an error. Functions were tested against invalid inputs, and the system was tested when an Internet connection was not available.

6. System Extension 1: Web Application

6.1. Design

The web application is composed of HTML5 and CSS3 files, as well as Dart applications responsible for providing an interactive web application and for creating the Polymer components *story-element*. The main and the *story-element* applications interact with each other to provide a responsive web application that switches between views, reacts to different user inputs and also displays notifications with useful system updates.

A *story-element* is a reusable Polymer web component that defines the structure and style of a story. A story has an ID (the same unique identifier used in the MySQL database), a type (Tweet or Google News story), a title (in case of a Google News story), a publisher, a date of publication, a headline, an image, a URL and a list of photos, when available.

In addition to adding animations and providing an interactive user experience, the main Dart application is also responsible for requesting stories from the server, which are then parsed by the same application responsible for creating the Polymer components *story-element*.

In terms of the design of the web application itself, it has a menu bar on top that indicates the current topic selected, and allows the user to search for

stories and load new stories. There is also an expandable menu on the left-hand side of the screen where the user can select a topic and filter stories by location, public reactions and stories that have been added to the blacklist.

Each story displays the profile image of the Twitter user or the image of the Google News story, the publisher, the date of publication, the title of the story (only applies to Google News stories), a short, descriptive headline, and buttons to read the full story, select a reaction and add the story topic to the blacklist. Tweets can also include photos, and in case there is more that one, the user can click on the current photo to switch to the next one.

6.2. Implementation

The web application was implemented using Polymer web components and according to the Material Design guidelines (Google, 2015). It should be simple, appealing and easy to use, while also being responsive and robust. It can also be used on mobile design

The main Dart applications allows the client to request stories from the server. If the server is offline or not responsive for over thirty seconds, a notification will appear to notify the user. The URL of the HTTP requests include the server's address and port, as well as the parameters topic and location, depending on the query.

After receiving a response from the server, the functions implemented in *story_element.dart* convert a JSON string into a list of Polymer components *story-element* with stories, and then each of the elements is appended to the browser's DOM and updated using data binding.

6.3. Discussion of Development Issues

The first issue that occurred during the development of the web application was that the client was unable to request resources directly from Twitter due to the restrictions imposed by most web browsers for security reasons. This is what initially led to the implementation of the proxy server on the server side, which could forward the client's requests by enabling the CORS headers.

There was another issue updating the *story-element*s in the DOM using data binding, which was solved by using the correct Polymer tag provided by the Polymer API for Dart. Some stories were also not displaying the photos due to broken URLs, as explained in Section 5.4.

Lastly, the stories with reactions were not being automatically removed from sections after the reaction had been updated or deselected, which was resolved by forcing the view update to occur only after the changes had been applied to the database.

6.4. Testing and evaluation

All the functions were tested individually and after integration with the rest of the system as they were being implemented. One of the goals of the testing procedures was also to ensure that exceptions were being caught and handled appropriately in order to prevent the system from failing in the event of an error. The system was also tested both without an Internet connection and when the server was offline or not responsive.

The HTML elements were tested with the developer tools available on the browser, in particular the application's appearance and behaviour in several mobile devices with different screen sizes.

7. Discussion

7.1. Achievements

News Anchor was developed in Dart and uses Polymer web components, following the Material Design guidelines. According to Google (2015), these are some of the tools that are helping shape the modern web and powering many of the web applications available today. They are relatively new languages and platforms that required learning new concepts and development practices, which will be useful for the future.

Working on the MSc Project in parallel with other activities also required a very good management of time, creativity and persistence to learn and produce a concrete product in a short amount of time and with a limited amount of resources, which turned out to be a positive challenge that helped to develop new organisational skills, an effective way of planning projects and efficient working habits.

It was not possible to implement all the ideas originally planned for the project, but News Anchor was still able to introduce new concepts which could be applied to a real-world application, as will be explained in the next section.

7.2. Novelty

Instead of implementing many of the features that can already be used in other services, News Anchor focused on introducing new concepts. News Anchor relies on Twitter and Google News to deliver news, offering good quality, professionally-written stories available on Google News, and also keeps the user up to date with the latest news posted on Twitter. It is often the case that stories are published on Twitter even before publishers have had time to write a full article about them.

In addition to this, anyone can post stories on Twitter, not only professional publishers. This concept is called social news, and the biggest advantage is that users can have access to stories through the perspective of other people and know more about their opinions. In terms of local news, this also means that it is easier and quicker to find out about what is happening in a certain location, as many people use their mobile phones to report events that are occurring round them. And all this is delivered in a single application to help users stay on top of the latest news in a convenient and easy way to use.

The other two features that differentiate News Anchor from other services available on the market are story reactions and a blacklist. Story reactions are an easy way to view the reactions other users have had to each story.. There are two available categories: positive reactions and negative reactions. In the positive reactions section, users will be able to find stories that most of the

other people have considered positive, and in the negative reactions section stories that have been considered negative. Any user is able to contribute to the public reactions by selecting between a positive and a negative reaction, and the more people use it, the better and more informative it becomes. Furthermore, implementation of user accounts in conjunction with reactions would allow for the collation of statistics regarding responses among certain demographics. For example a Conservative victory in the UK General Election may have garnered a mostly positive reaction amongst 45 to 60 year old British males, whereas amongst amongst 18 to 25 year olds of any gender the reaction may have been mostly negative.

The second feature is the blacklist. The blacklist behaves in a similar way to email spam filters, and allows users to filter out certain stories and topics from their news wall. If there is a specific topic that is currently being discussed and the user does not wish to read more about it, they can simply add it to their blacklist. These stories will still be available on the blacklist section and can be consulted at any time, and topics can be deleted from the blacklist at any time. Moreover, both this feature and story reactions are synchronised with the server in real-time and are available on any device.

7.3. Limitations and Further Work

One of the main constraints detected at the early stages of the project was caused by the diversity and different characteristics of multiple news

aggregators. Some of them, such as Twitter, provide their own API, and others such as Google News and NewsNow.co.uk only offer RSS feeds, each of them with different structures. Due to this, this project was focused on Twitter and Google News for the quality of its algorithms and for being used by many Internet users.

The goal of this project was to provide a simple aggregator of news aggregators, so it was less focused on the implementation of features that can already be found in most news websites and applications, and more focused on the implementation of the aspects that make News Anchor different from the other services. As a result, it does not offer features such as sharing stories on social networks, reorganising and managing the topics and sources of the stories, notifications, bookmarks, automatic detection of the user's location, or even many personal customisations.

Future development stages of the project should introduce the creation of user accounts and restricting the access to registered users only. This would allow users to have more control over the stories that they see, and also allow the implementation of more personal customisation features such as management of the topics and sources, and bookmark stories that can be read later even when offline. After this, users should also be able to take more advantage of story reactions, as they would be able to see the impact that a story has had on other people.

Additional features would include features that can already be found in several other services, in particular sharing news on social networks, automatic detection of user location, saving stories online to read at another time and notifications for breaking news and specific topics as configured by the user. In addition to this, users would also be able to create a profile where they could share stories and their opinions, and other users could send comment and initiate debates with them. Comments would also be available on the news wall, and these would equally be displayed on the user's profile.

At the moment only the latest fifty stories stored in the database are being displayed on the news wall. In the future the system should be able to load more stories progressively as the user scrolls down the page. Even though this was not implemented, the current implementation already provides functions that enable this to be done.

7.4. Final Conclusions

The development of an aggregator of news aggregators turned out to be harder than initially expected. There are multiple sources of news and each of them stores, organises and makes the data available in different formats, which increases the complexity of the problem. The original plan had to be reviewed and updated to allow a functional product to be delivered within the available time and with the resources available.

The analysis of related services allowed the understanding of their limitations and what the best practices are. Most of them provide short headlines with links to the original stories, filters by categories and search, with very few features and personal customisations. There are, however, services that provide features not offered by News Anchor such as sharing news, detection of user location, content adaptation and bookmarks, but on the other hand it introduces new concepts of filtering news with a blacklist and bringing a social environment to the application with news reactions. In addition to this, it implements the concept of social news by including Twitter stories in its sources.

The software was developed iteratively based on the spiral model, and each iteration implemented new requirements. The system is divided into core components and an extension, the web application. The core components comprise the proxy server and the crawler, and together they are the engine that retrieves new stories from Twitter and Google News, manages the database and delivers stories to the web client.

Overall, and despite having space for many improvements and future work, this project delivers a fully-functional application that, together with the research done, was able to demonstrate how much news aggregator services can benefit from social integration and more customisations without compromising simplicity and usability.

The whole implementation of the project consists of 3051 lines of code excluding configuration files.

8. References

Isbell, K. (2010) The Rise of the News Aggregator: Legal Implications and Best Practices. The Berkman Center for Internet & Society at Harvard University.

Boemh, B. (2000) Spiral Development: Experience, Principles and Refinements. Carnegie Mellon University.

Nieh, J., Yang, S. J. and Novik, N. (2000) A Comparison of Thin-Client Computing Architectures. Network Computing Laboratory, Columbia University.

Twitter (2015) Application-only authentication [online]. Available from https://dev.twitter.com/oauth/application-only [Accessed 11 July 2015].

Kesteren, A. (2009) Cross-Origin Resource Sharing [online]. Available from http://www.w3.org/TR/2009/WD-cors-20090317/ [Accessed 9 July 2015].

Twitter (2015) REST APIs [online]. Available from https://dev.twitter.com/rest/public [Accessed 8 July 2015].

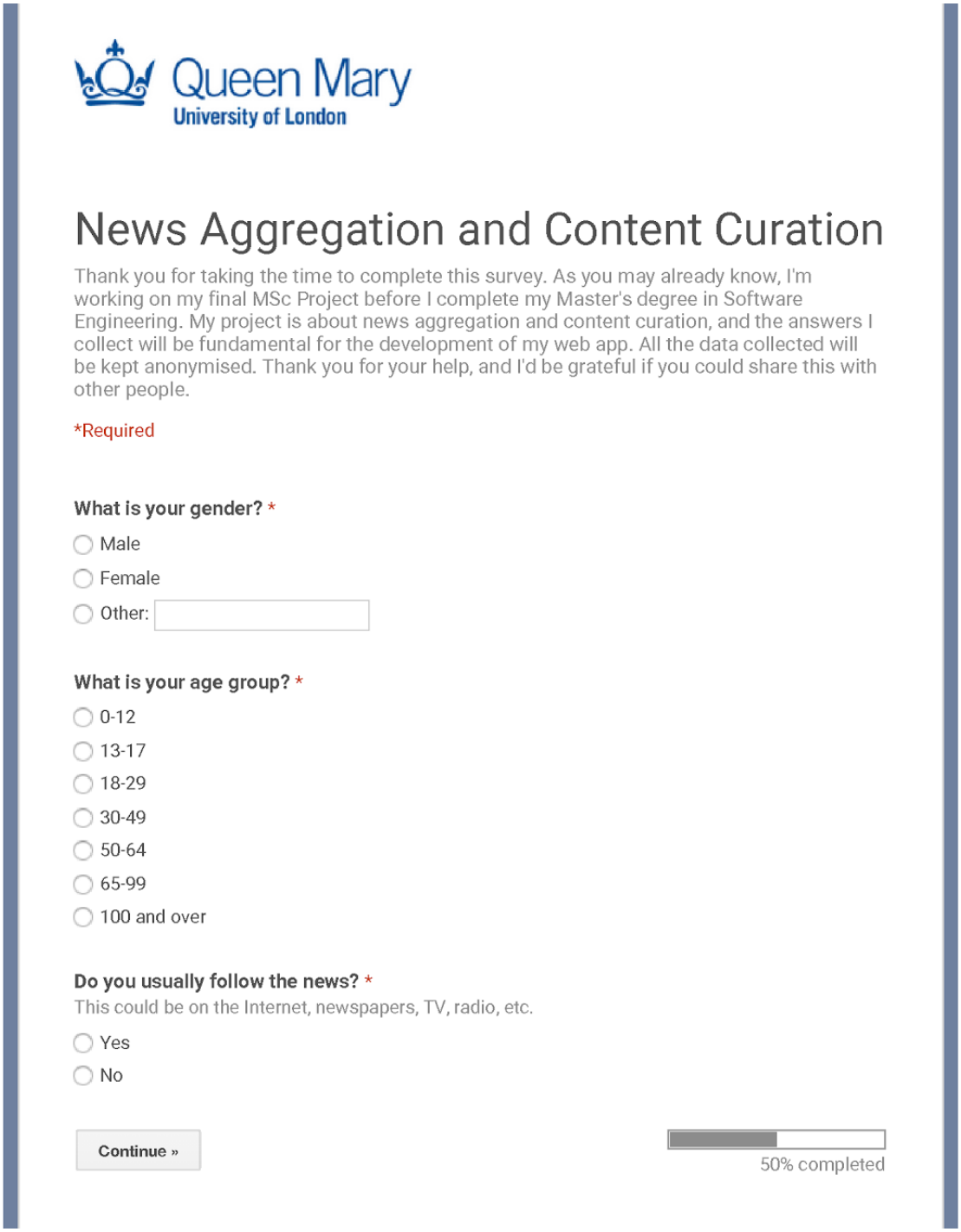
Google (2015) Introduction to the Material Design Guidelines [online]. Available from

https://www.google.com/design/spec/material-design/introduction.html [Accessed on 3 July 2015].

Dart (2015) Who Uses Dart [online]. Available from https://www.dartlang.org/community/who-uses-dart.html [Access on 21 August 2015].

Polymer Authors (2015) Polymer FAQ [online]. Available from https://www.polymer-project.org/1.0/docs/start/what-is-polymer.html [Accessed on 21 August 2015].

9. Appendix 1: User Requirements Survey



Queen Mary
University of London

News Aggregation and Content Curation

Thank you for taking the time to complete this survey. As you may already know, I'm working on my final MSc Project before I complete my Master's degree in Software Engineering. My project is about news aggregation and content curation, and the answers I collect will be fundamental for the development of my web app. All the data collected will be kept anonymised. Thank you for your help, and I'd be grateful if you could share this with other people.

*Required

What is your gender? *

- Male
- Female
- Other:

What is your age group? *

- 0-12
- 13-17
- 18-29
- 30-49
- 50-64
- 65-99
- 100 and over

Do you usually follow the news? *

This could be on the Internet, newspapers, TV, radio, etc.

- Yes
- No

Continue »

50% completed

Queen Mary
University of London

News Aggregation and Content Curation

*Required

User Requirements Survey

1. How do you normally follow the news? *

Tick all that apply.

- [ ] Physical newspapers/magazines
- [ ] Internet
- [ ] TV
- [ ] Radio
- [ ] Podcasts
- [ ] Other:

2. What is your favourite device to follow the news on the Internet? *

- ◯ Desktop or laptop computer
- ◯ Mobile device (mobile phone, smartphone, tablet)
- ◯ I don't follow the news on the Internet
- ◯ Other:

3. What is your default web browser on the desktop? *

This is the web browser that your use on your desktop or laptop computer for personal use. You can check what browser you are using here: https://whatbrowser.org/

- ◯ Google Chrome
- ◯ Firefox
- ◯ Safari
- ◯ Opera
- ◯ Internet Explorer
- ◯ I don't use any
- ◯ I don't know
- ◯ Other:

4. If you have a mobile device, which operating system do you use? *

This is the operating system on the mobile device that you use the most for personal use.

- ◯ Android
- ◯ iOS
- ◯ Windows Phone
- ◯ BlackBerry OS
- ◯ Amazon Fire OS
- ◯ Ubuntu Phone
- ◯ Firefox OS
- ◯ I don't use a mobile device
- ◯ Other:

5. What is your default web browser on your mobile device? *

This is the web browser on the mobile device that you use the most for personal use. You can check what browser you are using here: https://whatbrowser.org/

- ◯ Google Chrome
- ◯ Firefox
- ◯ Safari
- ◯ Opera
- ◯ Internet Explorer
- ◯ I don't use a mobile device
- ◯ I don't know
- ◯ Other:

6. What services do you use to follow the news? *

Tick all that apply.

- ☐ Specific newspaper websites (The Guardian, Daily Mail, etc.)
- ☐ Specific blogs
- ☐ Google News
- ☐ Twitter
- ☐ Facebook
- ☐ Google+
- ☐ Yahoo! News
- ☐ Flipboard
- ☐ News360
- ☐ Circa
- ☐ Pulse
- ☐ Google Play Newsstand (previously Google Currents)
- ☐ Feedly
- ☐ NewsBlur
- ☐ Digg
- ☐ I don't use any
- ☐ Other:

7. What features do you use in these apps or websites? *

Tick all that apply. If you select 'Other', please be as detailed as possible.

- [ ] Add/remove sources (news publishers)
- [ ] Add/remove topics and interests
- [ ] Search for news
- [ ] Find related stories
- [ ] Weather forecasts
- [ ] Detect user location to read local news
- [ ] Share stories on social networks
- [ ] Share stories within the app or website
- [ ] Comment stories
- [ ] Like and dislike stories
- [ ] Save/bookmark stories
- [ ] Read stories offline
- [ ] I don't use any
- [ ] Other:

8. In which media formats do you access the news? *

Tick all that apply.

- [ ] Text and images
- [ ] Video
- [ ] Audio
- [ ] Other:

9. How satisfied are you with the apps and websites that you currently use to follow the news? *

	1	2	3	4	5	
Not satisfied at all	○	○	○	○	○	Very satisfied

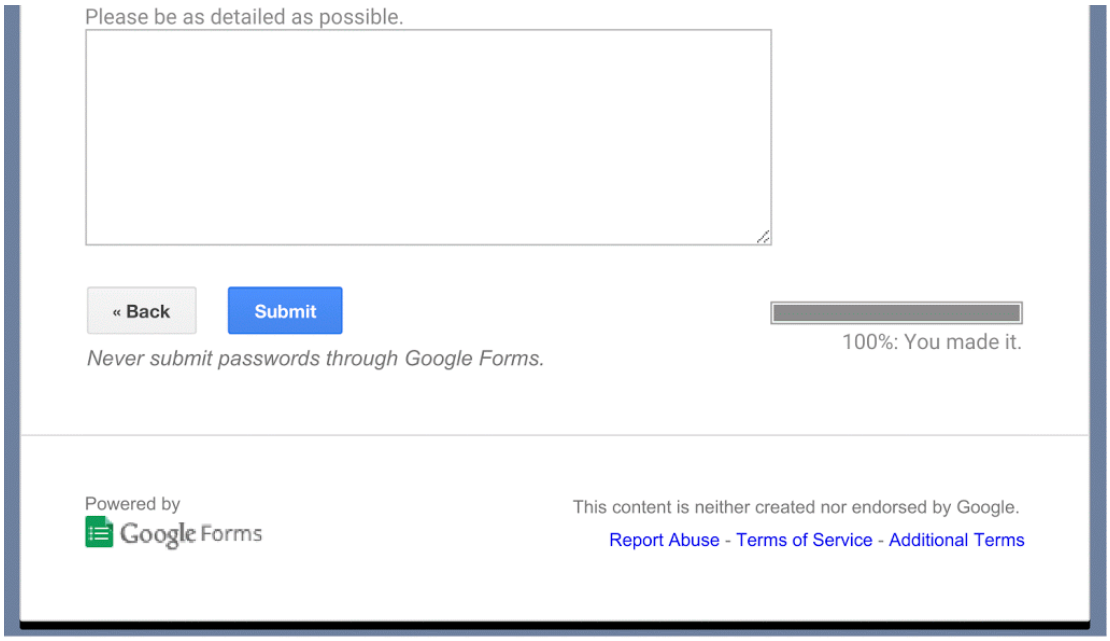
10. Would you be willing to use a new app or website to follow the news?

	1	2	3	4	5	
Not willing at all	○	○	○	○	○	Very willing

11. Would you be interested if it had a social network where you could add comments and share stories with other users? *

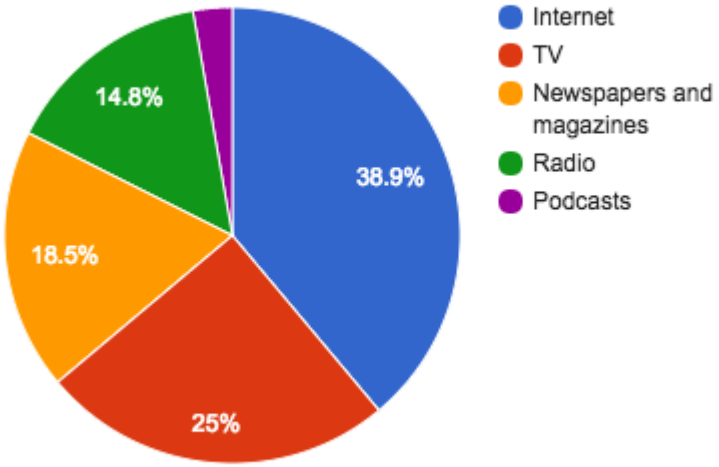
	1	2	3	4	5	
No, I don't need another social network	○	○	○	○	○	Yes, very interested

12. What features would you like to have in an app or website to follow the news? *

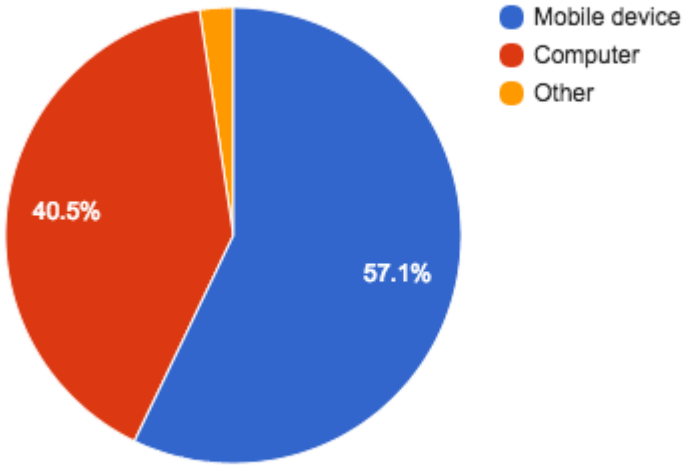

Please be as detailed as possible.
« Back
Submit
100%: You made it.
Never submit passwords through Google Forms.
Powered by
Google Forms
This content is neither created nor endorsed by Google.
Report Abuse - Terms of Service - Additional Terms

10. Appendix 2: User Requirements Responses

1. How do you normally follow the news?

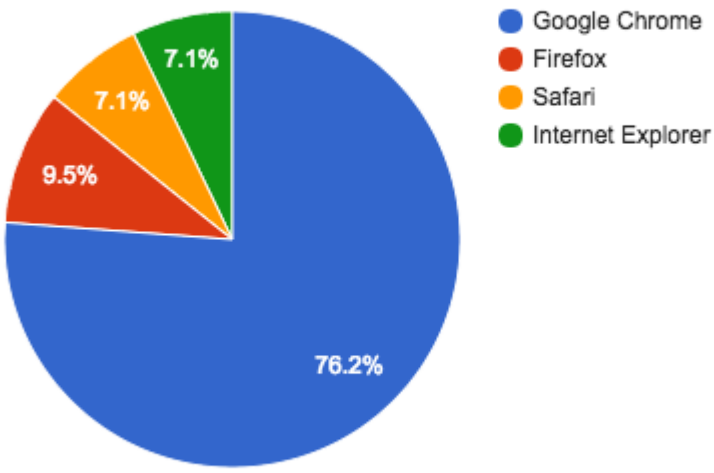


2. What is your favourite device to follow the news on the Internet?

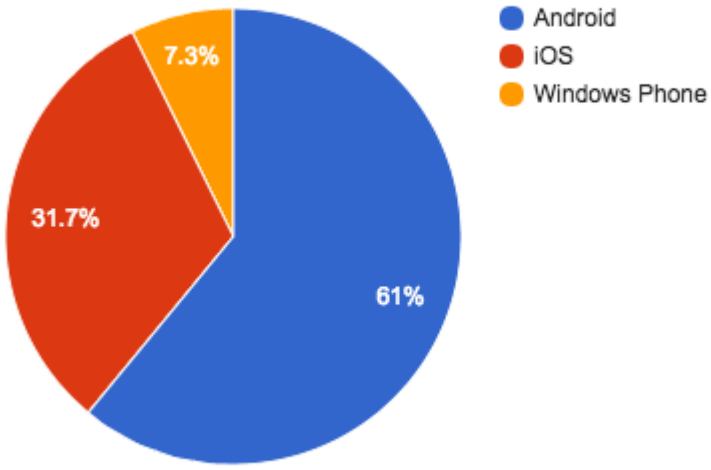


Others: Kindle.

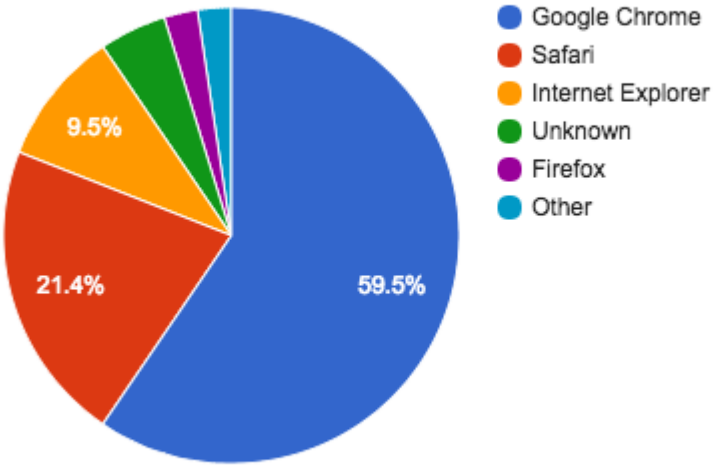
3. What is your default web browser on the desktop?



4. If you have a mobile device, which operating system do you use?

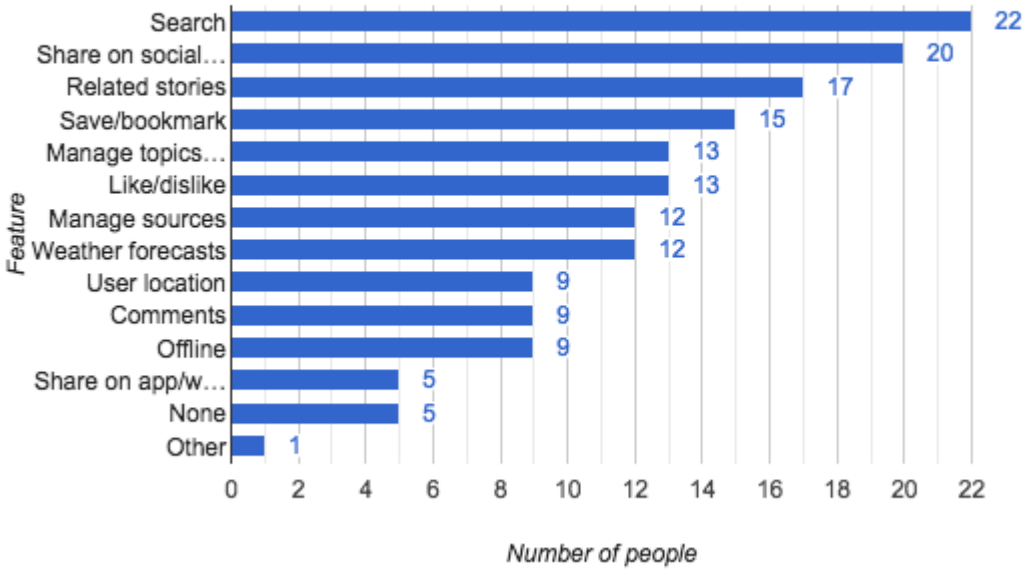


5. What is your default web browser on your mobile device?



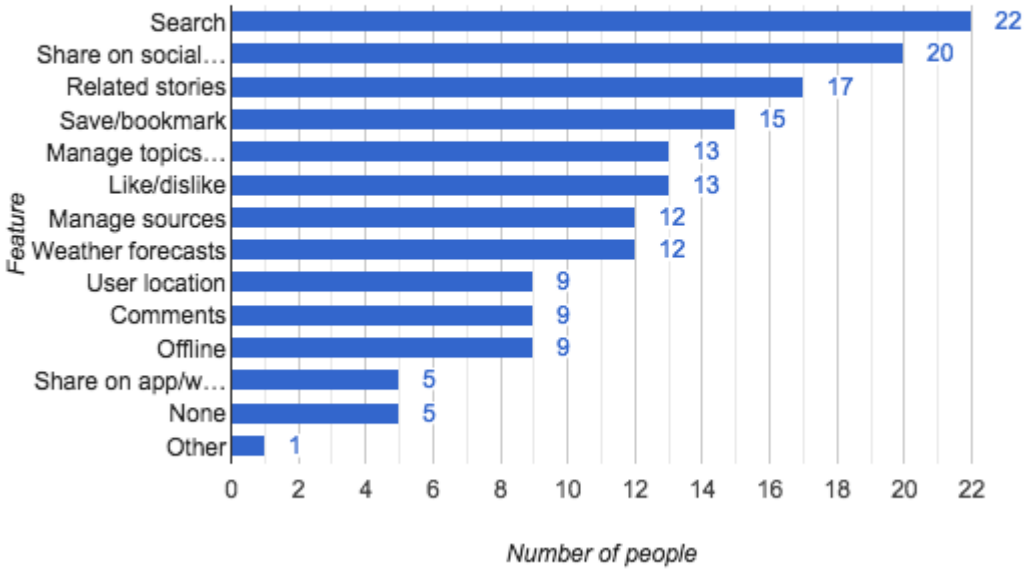
Others: UC Browser.

6. What services do you use to follow the news?



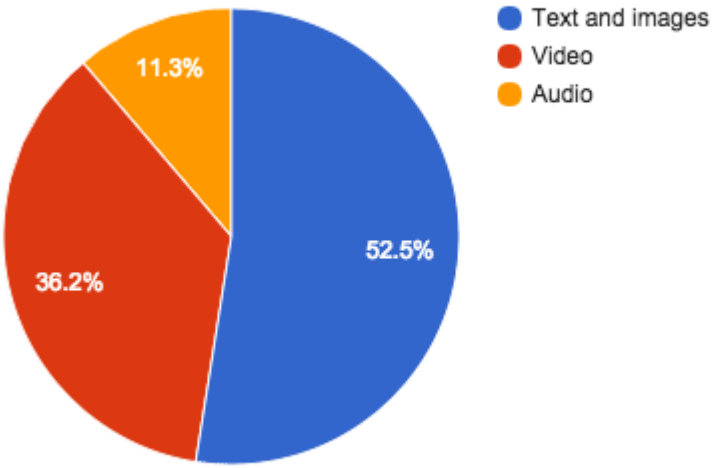
Others: Reddit, Hacker News, Slashdot and Samsung My Magazine.

7. What features do you use in these apps or websites?

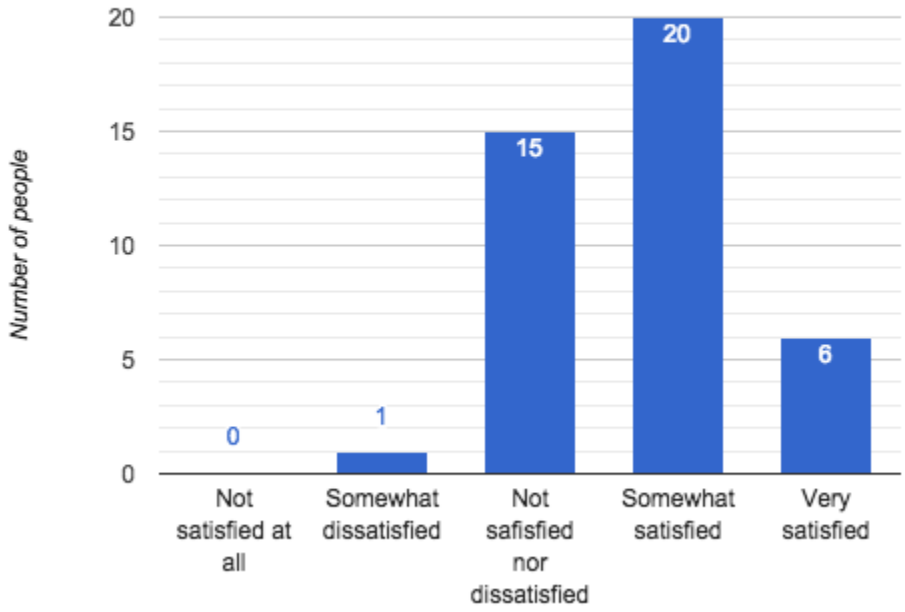


Others: Filter stories by criteria such as tags, title or author.

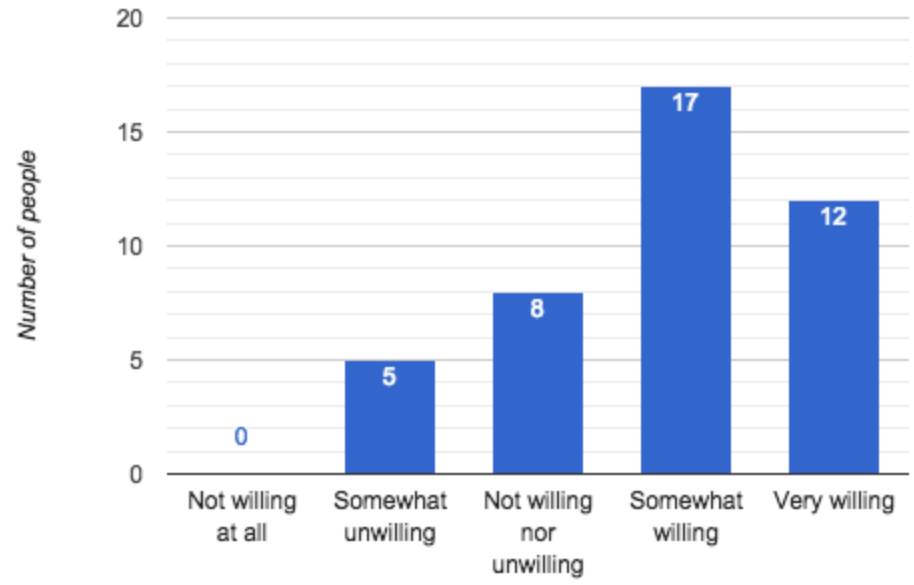
8. In which media formats do you access the news?



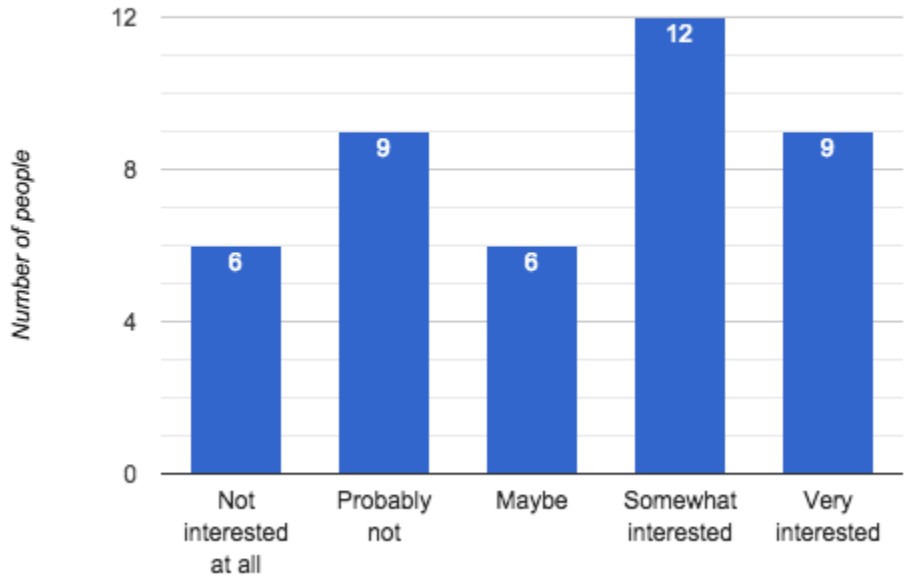
9. How satisfied are you with the apps and websites that you currently use to follow the news?



10. Would you be willing to use a new app or website to follow the news?



11. Would you be interested if it had a social network where you could add comments and share stories with other users?



12. What features would you like to have in an app or website to follow the news?

If the "social network" feature exists, it would be good if it could be set to only show shares, comments, likes and other interactions from my group of "friends". Another interesting feature would be a news source discovery by thematic, as many news sources are very specialised.
You could do a custom platform news, i.e. when a tragedy happens like in Nepal, I don't want to see more news about it because it doesn't interest me, and so I unfollow those news, and in the next 60 (X) days for example I won't see any news about Nepal. On the other hand if a story interests me enough (sports betting, schemes, frauds) I want to see more similar news in the next days even though they aren't usually popular news.
I think my favourite news apps are intuitive to use - the guardian for example. Other news apps I like are for the content, and are very simple. I like vice news for the content but hate that it tries to recommend other stories halfway through the current one.
Aggregation.
Notifications about breaking news worldwide and local (based on my

location). No ads or heavy graphics so that the news loads quickly. Well formatted for mobile devices i.e. screens smaller than a pc. A section for short interesting news e.g. an interesting picture with short summary as caption.
Quick to refresh, clean layout, no ads.
Smart and able to tailor the news to what I like and dislike. I should be able to say the keywords I don't like to read about. For example I don't like news about games so if I am given the option to filter out news about Xbox I'll be delighted.
It would be interesting to use my mobile device location to show me news that happened nearby me. All the other features already exists on other platforms.. so the option here is to do it BETTER than the others and add new features like the one I mentioned.
The app should learn what kind of news I'm interested, and afterwards show only relevant news corresponding to my interests. This process should be continuous so that it reflects the change of my interests.
The ability to push breaking news headlines through as a notification based on the user's preferences and not just based on what something, like the BBC, think is important.
The ability to comment on news stories. The ability to link to related stories.
Option to select news in selected genres.
To be able to comment on all stories not just selected ones as currently on the Guardian app.
Text, pictures & video.
I'd love to be able to turn off user comments. Sometimes I just don't want to look, then I do, and I am disgusted with ignorant people. I'd also love full-text traditionally written stories. I get irritated by this millennial-style writing–incomplete, un-referenced, incorrect. Sadly a lot of Internet writing lacks proper "who, what, when, where and why"'s.
Less advertisements through and around the articles.
I like to follow the latest news, though I never really use an app, but I always visit the websites of several newspapers. Now that I'm thinking, a nice thing would be to have several columns grouped by subject (imagining a desktop environment) although in an app it would not work as well.
Display of news summary.

Custom topics according to user settings.
Clean interface with easy to read and easy to scroll through the news.
I’d like an app that was simplistic but still had various options to comment, like, dislike, follow, unfollow, block, hide, ignore...
Country wise news sections as well as offline reading facility.
Stock price lookup, localization, balanced news sources and coverage of same event from all angles, share to social media.
Easy sharing to Facebook, links to news conferences or interviews.
I want constantly refreshed updates. Frankly, Twitter does everything that I want from a news service.
The ability to read a story as it is covered by different media outlets.
I would like to have news delivered to me without opinions or bias.
Easy navigation.
It would be great to be able to have a filter for positive news! I don't mean just articles from http://positivenews.org.uk/, but happy news stories. Reading offline would be cool, I don't know of any app where I can do that.
Mechanism to send me the most important news to my email.
One app that could be used to view all others.
A link to older news stories involving the same or similar topics, the people involved. Basically an archived stories button that would pull news that relates directly to the news at hand.
Text alerts.
Sharing capabilities.