



university of  
 groningen

faculty of mathematics  
and natural sciences

# Apollo

Simplicity and intuitiveness in a personalized  
multilingual reading tool

Bachelor's thesis

June 2017

Student: D. Chirtoaca

Primary supervisor: Dr. M. Lungu

Secondary supervisor: Dr. A. Lazovik

### **Abstract**

Learning a language is a long and exhaustive process. To encourage and help language learners study their language of choice, we analyzed, proposed, developed and evaluated a reading platform that is suitable for the above listed needs, allowing the users to interact with reading material for which they have interest in, with the ultimate purpose to make the learning process an enjoyable one.

The tool has been tested in a high-school environment, by a sizable group of French learning students. Important usability insights have thus been gathered that helped improving and fine-tuning the system, charting the road-map for future refinement opportunities.

# CONTENTS

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Structure	5
<b>2</b>	<b>Related Work</b>	<b>6</b>
2.1	Zeeguu Reader for iOS	6
2.2	Zeeguu Reader for Android	8
2.3	Amazon Kindle	8
2.4	The Effects of Extensive Reading	9
<b>3</b>	<b>Problem Description</b>	<b>11</b>
<b>4</b>	<b>Realization</b>	<b>14</b>
4.1	Article Scraping and Curation	14
4.2	Word Translation	15
4.3	Word Pronunciation	17
4.4	Translation Alternatives	17
4.5	Extended Features	18
4.5.1	Undo Translation	18
4.5.2	Text Selection	18
4.5.3	Article liking	19
4.5.4	Article starring	19
<b>5</b>	<b>Architecture</b>	<b>21</b>
5.1	Back-end	22
5.1.1	Flask app	22
5.1.2	Pre-processing	23
5.2	Front-end	23
5.2.1	ZeeguuRequests	24
5.2.2	Translator	25
5.2.3	Speaker	26
5.2.4	AlterMenu	26
5.2.5	Notifier	27
5.2.6	UndoStack	28

5.2.7	Starer	28	
5.2.8	UserActivityLogger	29	
<b>6</b>	<b>Results and Evaluation</b>		<b>30</b>
6.1	User Satisfaction	30	
6.2	Heat Maps	32	
6.3	Statistical Analysis	34	
6.4	Discussion	39	
<b>7</b>	<b>Conclusion</b>		<b>40</b>
<b>8</b>	<b>Future Work</b>		<b>41</b>
8.1	Dictionary support	41	
8.2	Preserve some visuals	41	
8.3	Preserve article state	41	
8.4	Particle verbs support	42	
8.5	Reading modes	42	
8.6	Reading rate	42	
<b>A</b>	<b>Workflow</b>		<b>43</b>
<b>B</b>	<b>Dependencies and Tools</b>		<b>44</b>
B.1	Flask	44	
B.2	Newspaper	44	
B.3	Hotjar	44	
	<b>Bibliography</b>		<b>45</b>

# INTRODUCTION

---

We are living in multicultural societies. Travelling, migrating or simply commuting, eventually makes us interact with a variety of individuals. To be able to adapt and integrate into these new societal contexts, communication is one of the primary tools that we use. The most meaningful and efficient communication can be achieved through the means of language. Hence, proficiency in multiple languages gives us a distinct advantage in the current globalized world that we live in.

What we conclude from the above is that language learning is becoming an ubiquitous and even natural process when dealing with day-to-day and human-to-human interactions. Facilitating this process can thus benefit a large part of the society.

One of the most beneficial ways for a learner to improve their mastery of a foreign language has proven to be reading. In fact, McCarthy [3] argues that extensive reading serves as a comprehensive training ground that has the potential and ability to improve the other crucial language skills as a result: writing, listening, and speaking. Moreover, this same idea of extensive reading is backed by research, conducted by a large number of researchers and educators.

Several empirical studies and syntheses on the topic have concluded that indeed extensive reading does have a positive impact on the above mentioned points. Renandya [7] argues in his article about "The Power of Extensive Reading" that the benefited gains from extensive reading make it an attractive method to be encouraged by language teachers. It should not be considered as a replacement of the usual intensive reading, but it must be acknowledged that it allows the student to focus on broader intricacies of the language he or she is studying. Moreover, Namhee [4] presents research results in which the significant impact of extensive reading over reading comprehension, reading rate and vocabulary acquisition is highlighted (see section 2.4).

The state of the art in supporting readers of foreign languages is however lacking. On one hand, traditional learning materials in textbooks are most often unattractive as they target the non-existent generic learner. On the other hand,

the majority of the materials available online are usually incompatible with the learner's current language skill level.

Making reading more adapted to the personalized needs of the individual will help learners become more engaged, and thus enable them to progress easier and faster. Language learners would become more committed if they could read content that they find appealing. Moreover, some learners are often trying to maintain a parallel learning process of multiple different foreign languages.

The issues concerned with providing the user with an interface through which sources for articles in their target language(s) can be identified and subscribed to, as well as listing the available reading options, is analyzed in detail in the related thesis - Prometheus<sup>1</sup>, authored by L.A.H. van den Brand [10].

This thesis however, is focused on creating and evaluating a reading platform that provides users with a straightforward reading experience, which is characterized by the ability to read the material they find appealing in an unobtrusive manner, such that the clutter they have to face on usual websites, is removed, leaving the user with clean and efficient text interaction. At the same time, offering the availability of in-place translation possibilities, but as McCarthy [1] stresses, such activity has to be discouraged in order to push the learner to concentrate on the message conveyed rather than the meaning of a particular word or phrase thus facilitating the premise of extensive reading and ensuring a pleasant experience.

The research questions that this project analyzes and tries to answer are:

- RQ1** (a) What is a minimally required User Experience that allows for non-intrusive, extensive reading?  
 (b) What is the simplest software architecture that would support it?
- RQ2** Would a system like the one described in this proposal be helpful for the learners? Would it increase the engagement of the users with respect to traditional textbook materials?

---

<sup>1</sup> Where Prometheus is the god of forethought, Apollo is the god of analysis.

## 1.1 STRUCTURE

This document presents and describes the reading environment of the Zeeguu Reader, together with the results of the user study and the evaluation procedure. The rest of this document is structured in the following manner:

2. **Related work:**

This chapter presents relevant work related to the problem at hand.

3. **Problem Description:**

This chapter focuses on the derived requirements for the reading platform.

4. **Realization:**

This chapter demonstrates the implemented functionality and features of the system (i.e. the end product).

5. **Architecture:**

This chapter is concerned with the architecture, design and implementation of the system.

6. **Results and Evaluation:**

This chapter analyzes the obtained results from user testing and evaluates those accordingly.

7. **Conclusion:**

This chapter draws conclusions and general insights from the project as a whole and determines its level of success.

8. **Future Work:**

This chapter lists features and changes to the system to improve and further iterate on.

**Appendix:** This section provides more information about the project's development such as workflow, used dependencies, tools and others.

## RELATED WORK

---

Even though this project is built from the ground up, there have certainly been influences with respect to design decisions as well as analysis of necessary or absent features. Moreover, the existence of the Personalized Multilingual Reader is partially the result of aiming to solve the issues with the related platforms.

### 2.1 ZEEGUU READER FOR IOS

The Zeeguu Reader for iOS, designed and developed by Jorrit Oosterhof [5] is a nicely designed application that provides the user with a sufficiently streamlined reading environment (Figure 1), that serves its purpose well.

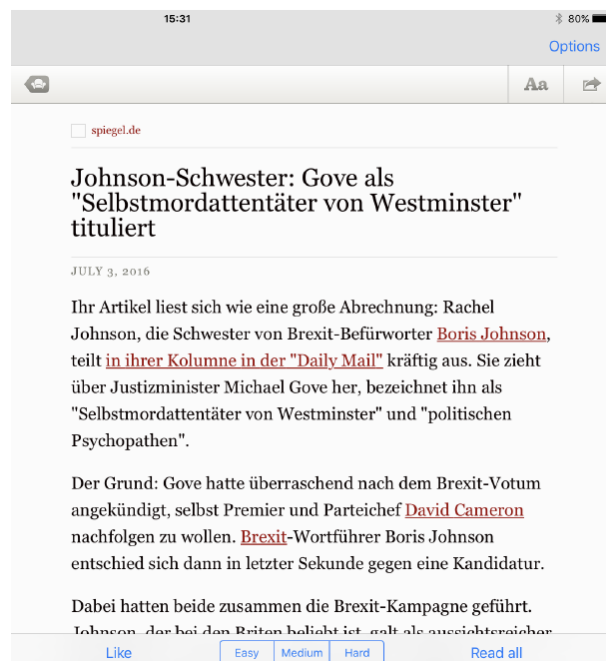


Figure 1: Reading an article in the iOS Reader



It allows the user to obtain translations by tapping the words. Moreover it offers the option to switch between a few translation modes:

1. Single word.
2. Pairs of words.
3. Sequence (interval) of words.

Translations are inserted into the text, but these cannot be removed, which sometimes proves to be a nuisance.

Additionally, it supports changing and adapting the font size to the reader's needs and disabling the links within the article. Last but not least, the person reading will obtain a pronunciation for the words for which a translation has been requested. However the peculiarity of this feature is that either all the translated words are pronounced or none of them, since the text-to-speech functionality is turned on by a menu toggle. The above listed options can be easily identified in the figure below:

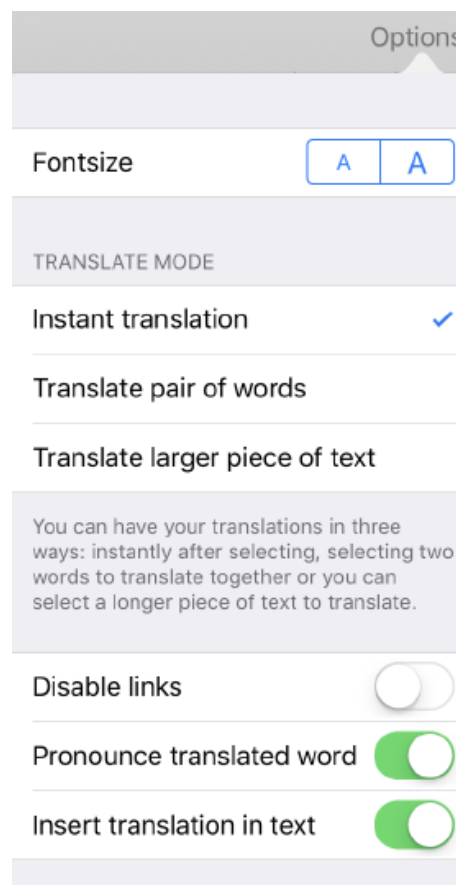


Figure 2: Article options in the iOS Reader

## 2.2 ZEEGUU READER FOR ANDROID

Prior to the iOS reader, an Android solution was built by Linus Schwab [8]. A great application, but a completely independent and separate project from the iOS counterpart.

It uses Feedly <sup>1</sup> as its RSS feed aggregator, a rather popular solution at the time and even more so today. It thus provides a neatly stylized interface for source and article finding. It uses an extended (custom) WebView client for the reading environment, but translation functionality is rudimentary and even cumbersome to use. This is due to it using the native selection of the WebView client, requiring the user to press and hold over the desired word for an extended time. Moreover, expanding a selection is handled by manually adjusting the selection hooks. Last but not least, translations are transient, displayed temporarily as a tooltip, thus requiring the user to re-request them if he or she finds the need. The figure below summarizes the above listed aspects:

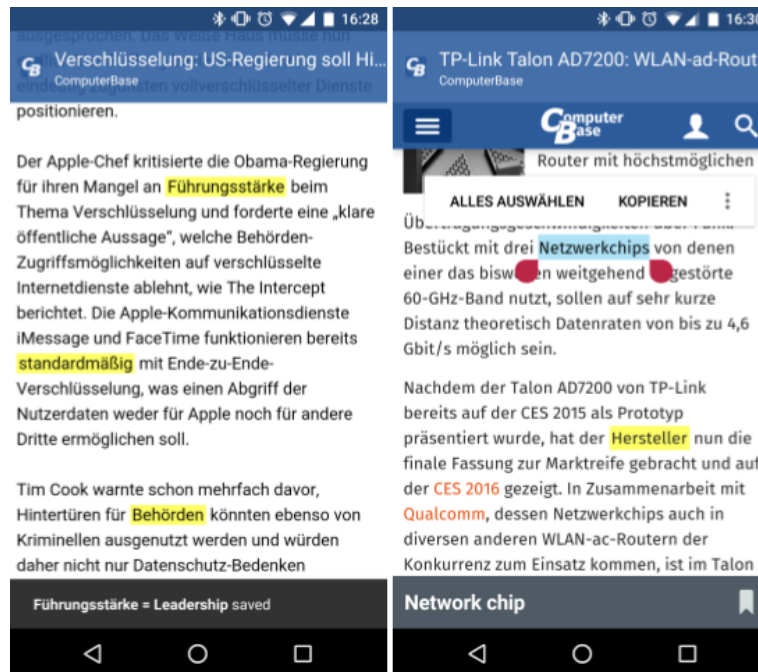


Figure 3: Reading an article in the Android Reader

## 2.3 AMAZON KINDLE

The reading platform provided by Amazon, is a very extensive one, with plenty of reading material choices. Together with a dictionary (Figure 4) extension (from the target learning language to the reader's native one), this can be considered a viable language learning platform since it also tracks the words

<sup>1</sup> <https://developer.feedly.com/>

that are being translated (i.e. Vocabulary Builder), on which the person reading can later practice <sup>2</sup>.

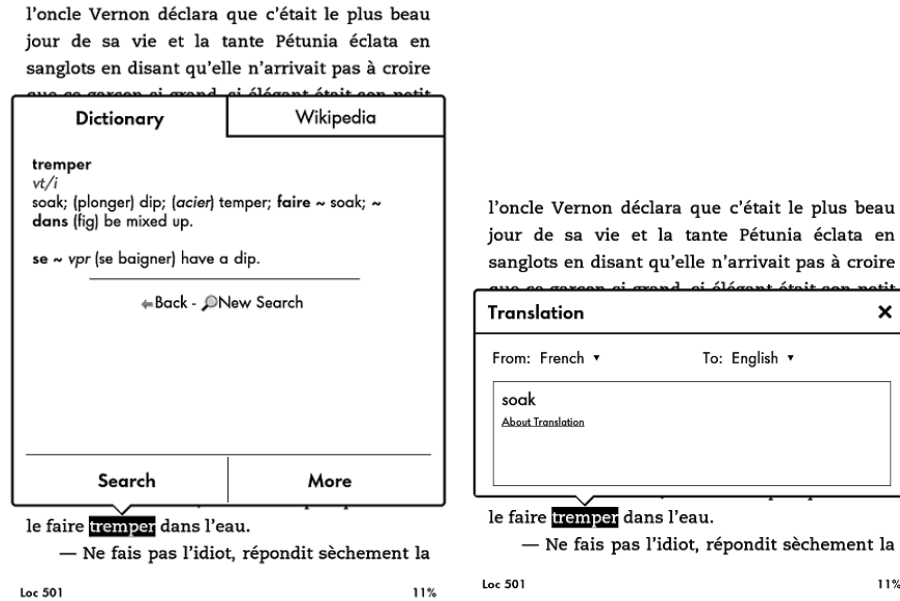


Figure 4: Defining and translating words on Amazon Kindle

## 2.4 THE EFFECTS OF EXTENSIVE READING

To elaborate on the previously introduced research on extensive reading [4], we should underline that it argues in favor of the benefits of extensive reading towards:

- reading comprehension
- reading rate
- vocabulary acquisition

The research had compelling conclusions, one of which can be summarized in Figure 5, extracted from the research paper:

<sup>2</sup> <http://www.makeuseof.com/tag/learn-language-using-kindle-paperwhite/>

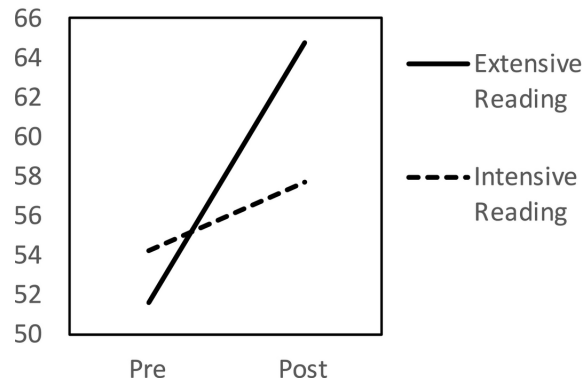


Figure 5: Vocabulary Mean Scores Across Group and Time

What we notice is that the group of students who performed extensive reading, caught up and even surpassed the intensive reading group, leading to the conclusion that the potential for better vocabulary acquisition lies with the former reading methodology. Furthermore, similar results apply to the other two topics of research: reading rate and comprehension.

The above are just some of the available solutions that partly solve our problem. Each of them have specific advantages and/or features, but neither seems to be attaining the goal of intuitive and simplistic multilingual reader.

The next section underlines the weaknesses of the above listed applications and sets up the main problem description aspects, while focusing on the idea of extensive reading and its impact on language learning.

## PROBLEM DESCRIPTION

---

After analyzing the related work, presented in the previous section, certain conclusions have been drawn. There are certain very good ideas bound to these projects that we would like to inherit and build upon:

1. The simple interaction mechanism from the iOS Reader that allows for retrieving a translation simply by tapping a word.
2. The ability to pronounce words similarly easy.
3. The possibility to check out alternatives for a translation.
4. The ability to extend a translation for more than one word as in the Android Reader.

Nevertheless, the listed platforms are burdened by several shortcomings. In return, this product tries to overcome a handful of them:

1. Have one platform that is available on multiple devices from different ecosystems. The iOS and the Android reader have little to nothing in common, thus switching platforms would require the user to adapt to a new interaction model.
2. The iOS reader depends on the already offline readability tool <sup>1</sup> and the Android one uses two separate APIs to implement its functionality - Zeeguu and Feedly. This is undesired and comes with multiple inconveniences: the iOS app is no longer functioning while the Android counterpart requires the management of two accounts, associated with the two used APIs.
3. Intuitive interface for translating words, without the need to switch between several modes or to access hidden interface components.
4. Better handling of the text-to-speech functionality.
5. Even better article scraping and curation.

---

<sup>1</sup> <https://www.readability.com/>

What concerns more concrete specifications of the actual features that the system should be equipped with, we are listing the following core feature set:

1. The system should be able to extract the textual content given an article URL, curate it and present it into a readable, non-invasive format such that any unnecessary clutter is no longer present, while not depending on third party services.
2. The user should be able to translate (obtain a translation for) words with the simple action of tapping (clicking) on a word, to allow for easy and efficient text interaction. The difficult mechanism from the Android version or the different modes from the iOS version are not efficient enough to not break the user's reading flow.
3. In order to enhance the language learning process, the user can directly obtain a pronunciation for the translated word or sequence of words. The pronunciation should be handled on top of the translation, unlike the iOS version where the same interaction event triggers both of them.
4. Taking into consideration that machine translation cannot be 100% accurate, the user can supply a translation suggestion of his or her own, thus opening the possibility to improve translations.

From here on, by integrating the primary goal of this system - stimulating extensive reading, the main non-functional requirements have been identified to be connected with the following:

1. The system should be web based in order to allow for cross-platform use, with seamless compatibility across all ranges of devices (from desktop computers and laptops to tablets and smartphones).
2. The system's interface should support proper scaling such that it adapts to the size of mobile device environments, since it is one of the most used platforms nowadays.
3. The system should be highly maintainable, such it can effortlessly be adapted to the user feedback or to new feature sets. Equally, high quality code with a high level of modularity and detailed documentation, represent a significant aspect in achieving that.
4. The system should be focused on delivering high usability, which is determined by intuitiveness of the user interface and simplicity in the user interaction. There should practically be no learning curve and the reading environment should not contain any distractions.
5. The system has to perform swiftly, to give the impression of seamless interaction and to supply the user with visual feedback that informs him or her about an ongoing activity.

These non-functional requirements serve as a foundational guideline in the development process of the personalized reader, to which we abide.

This platform is built around the functionality of the Zeeguu API <sup>2</sup>. Most of the features previously mentioned will be built on the mentioned API, by accessing and directly utilizing its provided resources, among which we can mention user account management and translations provider.

---

<sup>2</sup> <https://github.com/mircealungu/Zeeguu-API>

# REALIZATION

---

To describe the solution we will start by looking at individual parts of the system and how these parts solve the problem at hand. The core features, which have been described in the previous section, are explored below by taking a look at them in their implemented state. Moreover, this section is extended with a set of additional features that complement the core feature set.

## 4.1 ARTICLE SCRAPING AND CURATION

One of the main problems that we encounter while browsing the web is the huge amount of clutter and distractions that accompany the information that we are looking for. Therefore, reading articles tends to become more about dodging the annoyances than about the time spent in a pleasant manner. To solve this problem, the system performs an intelligent and thorough cleaning process for a given article, in its original web format, leaving us with pure, meaningful textual content.

Among the things that the system filters out as clutter we can enumerate the following:

1. banners
2. advertisements
3. pop-ups
4. UI of the website
5. links
6. headers and footers
7. pictures and videos
8. unconventional font styles (replaced by uniform and readable font style)
9. any other references that have little to no relation to the actual content.



The transformation is rather profound and it can best be evaluated by taking a look at what it achieves, given an article page to be processed (Figure 1).



Figure 1: Original website vs Reading environment

The visual improvement is noticeable, allowing for considerably larger font, while still being able to fit more than 60% of the content of the original page. The person reading can now focus on the content of the article without having to constantly avoid the extra material displayed on the page.

## 4.2 WORD TRANSLATION

One of the main reasons for this platform's existence is its ability to give in place translations. In response to that, a lot of time was dedicated to develop something that works seamlessly without interrupting the reading experience by either obstructing the reading environment or requiring to manipulate the text in any manner whatsoever. When the person reading has troubles understanding a word, tapping on it retrieves its translation (Figure 2).

La **vicepresidenta** **Vice president** del Gobierno, Soraya Sáenz de Santamaría, ha advertido este martes ante la pretensión de los soberanistas catalanes de aprobar una ley que en 48 horas permita la declaración de la independencia, que "al Estado le bastan 24 horas para recurrirla y obtener su paralización".

Figure 2: Obtaining a word translation

The user can chain a few consecutive words into a single translation if it's either a phrase that's new, or if the initially obtained translation needs more words combined with it, to deliver the correct contextual meaning. Again, everything works by tapping the words, as consecutive words will be automatically merged into a 'translation bubble', unifying the entire interaction mode to simple taps on the words (Figure 3). Moreover, this minimalistic interaction model serves another purpose - it is easy to use for words and phrases, but it discourages users to translate entire sentences or even paragraphs, following McCarthy's [1] idea presented before - extensive reading should discourage intensive use of translations.

La **vicepresidenta del** **Vice president of** Gobierno, Soraya Sáenz de Santamaría, ha advertido este martes ante la pretensión de los soberanistas catalanes de aprobar una ley que en 48 horas permita la declaración de la independencia, que "al Estado le bastan 24 horas para recurrirla y obtener su paralización".

La **vicepresidenta del** **The Vice-President of the** Gobierno, Soraya Sáenz de Santamaría, ha advertido este martes ante la pretensión de los soberanistas catalanes de aprobar una ley que en 48 horas permita la declaración de la independencia, que "al Estado le bastan 24 horas para recurrirla y obtener su paralización".

Figure 3: Obtaining a word sequence translation

Retrieving translations instantly would indeed be an ideal ability of the system. However, we have to accommodate for the fact that in reality this is not always achievable. Hence, to compensate for any delays the API might experience, the user is presented with a subtle, yet clear, animation, that distinctly conveys the message that the results will arrive shortly. Not only that, but as we can see in Figure 4, the system tries to approximate the length of the to be received translated word, by inserting 'dots' in the translation field, to inform about content that will be added at that location.

La **vicepresidenta del** **The Vice-President of the** Gobierno, Soraya Sáenz de Santamaría, ha advertido este martes ante la pretensión de los soberanistas catalanes de aprobar una ley que en 48 horas permita la **declaración** ..... de la independencia, que "al Estado le bastan 24 horas para recurrirla y obtener su paralización".

La **vicepresidenta del** **The Vice-President of the** Gobierno, Soraya Sáenz de Santamaría, ha advertido este martes ante la pretensión de los soberanistas catalanes de aprobar una ley que en 48 horas permita la **declaración** **declaration** de la independencia, que "al Estado le bastan 24 horas para recurrirla y obtener su paralización".

Figure 4: Waiting for a translation

### 4.3 WORD PRONUNCIATION

A very useful feature when studying a language, is the ability to perform text to speech on the newly encountered words. Even though the current state of text-to-speech engines is still in its development (improvements in this domain are rapidly evolving), it proves to be a valuable addition to the system.

As of right know, the languages that are properly supported include: German, Dutch, Spanish and French. These are also the languages in which articles can be read. Support for more languages should and will expand with time.

Once something is translated the original word is highlighted with a subtle shade of blue. Clicking on it makes the computer pronounce it.

### 4.4 TRANSLATION ALTERNATIVES

Machine translations are not always fully accurate. Sometimes the meaning for something can only be deduced from the context, other times it is part of a group of words that determine the meaning. Moreover, there are cases in which even though a translation is appropriate, it does not convey what was meant. That can be due to that word (or word combination) be a replacement for something else, more specific (e.g. l'hexagone in French actually refers to France), that is culturally not semantically embedded into the language.

To our help, comes the ability to view a list of alternatives, as a dynamic drop-down menu. It is accessed by a click on the translated part of a 'translation bubble'. From here, replacing the original translation with one of the available alternatives is done by clicking on the preferred and best suiting alternative (Figure 5).

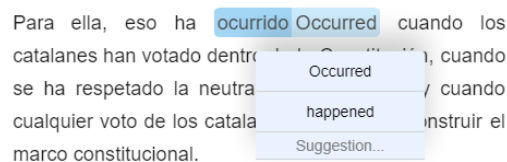


Figure 5: Translation alternatives

If however neither of the provided options are satisfactory enough, the user is encouraged to suggest his or her own alternative. A dedicated field at the bottom of the alternatives menu will allow them to do just that. The system will then use the suggested translation instead (Figure 6).

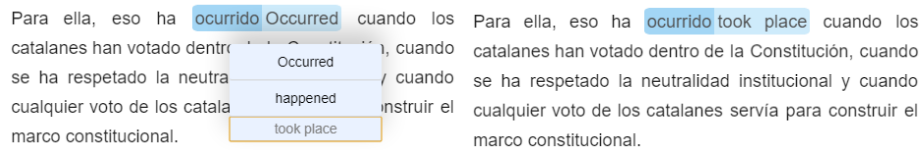


Figure 6: Suggesting a translation

## 4.5 EXTENDED FEATURES

Developing, testing and receiving feedback is what eventually helps determining not only the faults in a platform, but also its needs. Working in iterations is what granted us the ability to understand what works well and what is still in need of improvement. Every release version delivered improved core features while the process itself allowed us to make certain observations, both from development testing and user testing, which in turn have been summarized in a few feature definitions.

### 4.5.1 Undo Translation

Making the translation mechanism so simplistic and intuitive, results in a few small issues. One of them is the fact that a user could accidentally translate a piece of text without actually willing to do so. To solve this problem, the undo feature was introduced to the system (Figure 7).

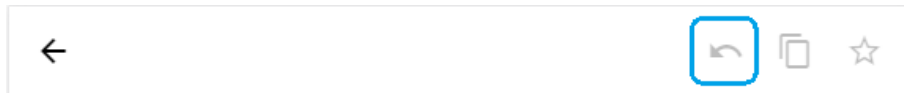


Figure 7: The 'Undo' button

The way it works is very straightforward: after translating something, click the undo button and the last action will be undone, in this case the last translated word will be restored to its original state.

Moreover, if two consecutive words are translated in succession quick enough, the system will consider that as a unique action. This, in turn, results in the undo system affecting both words and restoring them.

### 4.5.2 Text Selection

In order to streamline the translation interaction, there was one necessary trade-off - disabling text selection. This does not interfere with the user while they are enjoying an article, it actually only enhances the stability and fluidity of the translations system. If however they feel the need to copy, save or google some

text within the article, selection needs to be enabled. Deriving from a similar feature on the iOS Reader [5], a simple toggle button is placed in the top right corner of the page to provide the user with that ability (Figure 8).

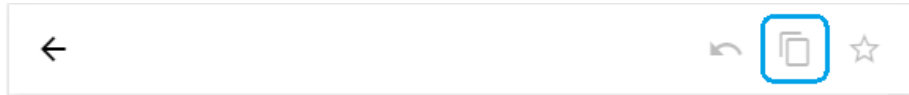


Figure 8: The 'Copy' toggle

After toggling it, the text interaction is the good old click and drag to select.

#### 4.5.3 *Article liking*

In the era of social media, people have become so accustomed to virtually stating that they like something, that having a 'like' button is becoming an almost ubiquitous thing.

At the end of each article, after a good read, the user encounters the like button. It of course works just as expected (Figure 9).



Figure 9: Liking an article

A feature like this is actually of great value. In the long run, it can be used to determine the interests a user has. Based on that, we can then provide him with suggestions to read. Besides that, we can evaluate the quality of an article source, deduced from the amount of likes its articles receive on an usual basis.

#### 4.5.4 *Article starring*

Sometimes, an article is too long to be read in one single reading section. Some other times, an article might simply be worth coming back to, both from the perspective that it presents an excitingly interesting story, or it seems like a very good article on which to practice the corresponding language. Both of these needs are fulfilled by the starring feature.

The 'star' toggle can be easily identified in the top right corner, on the action bar (Figure 10).



Figure 10: Starring an article

It gives visual feedback upon starring something, by filling up the star icon's background. Having the star always in the field of view ensures that a user can easily star the article and move on.

## ARCHITECTURE

---

The reader is built to run in the web browser. The primary architectural decision that guides the development of this platform is to build the server-side (back-end) as a Python Flask app and the client-side (front-end) in ECMAScript 2015 (ES6). The fact that our back-end is particularly lightweight is accommodated by Flask in a very simplistic and compact manner. What concerns ES6, since it currently is still bound to limited support across browsers, it is transpiled into compatible JavaScript code, bundled into one file, using Webpack.

The user interface is primarily dominated by the Material Design Lite framework. As its name suggests, it is used in order to abide by the material design principles. MDL has the ability to correctly and automatically scale to a large variety of resolutions. Hence, it considerably improves the cross-platform support that this project aims for.

In the upcoming subsections, we take a look at each one of these components and analyze their structure and functionality, starting from the back-end and then continuing with the front-end.

A glimpse of the platform's architecture overview can be seen in Figure 1. The related subscription management system (i.e. Thesis Prometheus [10]) is treated as a black box in this schema.

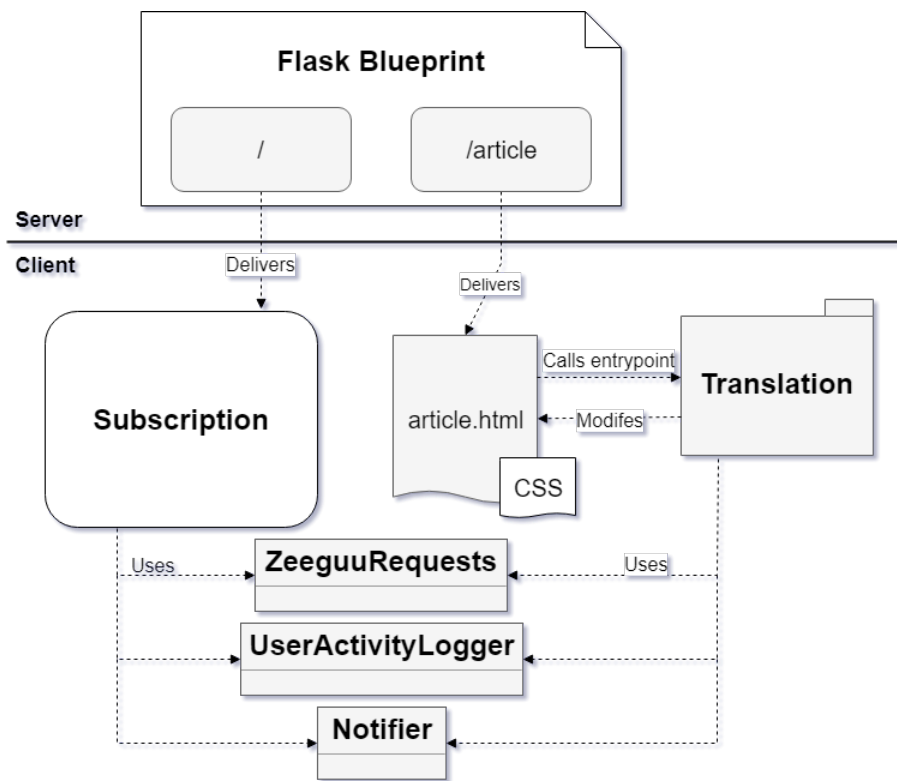


Figure 1: System overview

## 5.1 BACK-END

The Flask server, upon successful validation of the user's `session ID`, redirects to the `article.html`. This HTML document references the packaged script to be executed. It is also the place where the actual content of the article is appended.

### 5.1.1 Flask app

What concerns the back-end functionality tied to the reading platform, the Flask app, which is also a Blueprint, contains the endpoint `/article`, that responds to a GET request. It expects the caller to supply the URL to the article to be preprocessed, as a GET parameter. Additionally, the language of the article can be supplied, otherwise it is inferred by the article extracting tool, as well as a boolean parameter to assert whether this represents an article from the starred list of the user.



### 5.1.2 Pre-processing

Previously we have demonstrated what our system achieves through article curation. Underwater, this is mainly accomplished with the help of a Python tool - Newspaper3k and it occurs in the following order:

1. download the article page (entire html page)
2. parse the page
3. extract the content
4. wrap words with custom tags
5. insert content into the template

Newspaper [6] allows us to simply feed it the URL to the article of interest, additionally (optionally) the language of the article, and then extract the meaningful textual content from that web page, separated into title, body, authors, etc. (for more information on Newspaper see appendix B).

From here on, we construct the page template, by inserting the retrieved content at its appropriate location. Also, a crucial part in the functionality of the reader is played by the wrapping of words into custom tags (i.e. 'zeeguu' tag). This is an improved idea borrowed from the iOS Reader, that eventually eases the manoeuvrability of words on the front-end, which will be explained later.

To accomplish this however, we are using an extended regular expression (Figure 2) to identify individual words. It is constructed based on the list of unicode characters <sup>1</sup>, characters identified in the Indo-European languages, to which we have added special characters such as the hyphen, used in compound words (e.g. non-stop) and the apostrophes, used in article - word combinations (e.g. l'animal). By doing this, we are certain (up to a degree) that words are properly separated and isolated.

```
word = "([a-zA-Z0-9\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u00FF\u0100-\u017F\u0180-\u024F_'\"]+)"
```

Figure 2: Regular expression for identifying words

## 5.2 FRONT-END

To offer an overview of the architecture of the reading platform, we provide the following schema that shows how the functionality is divided among ES6 classes, which allows us to approach the problems at hand in an OOP manner:

<sup>1</sup> [https://en.wikipedia.org/wiki/List\\_of\\_Unicode\\_characters](https://en.wikipedia.org/wiki/List_of_Unicode_characters)

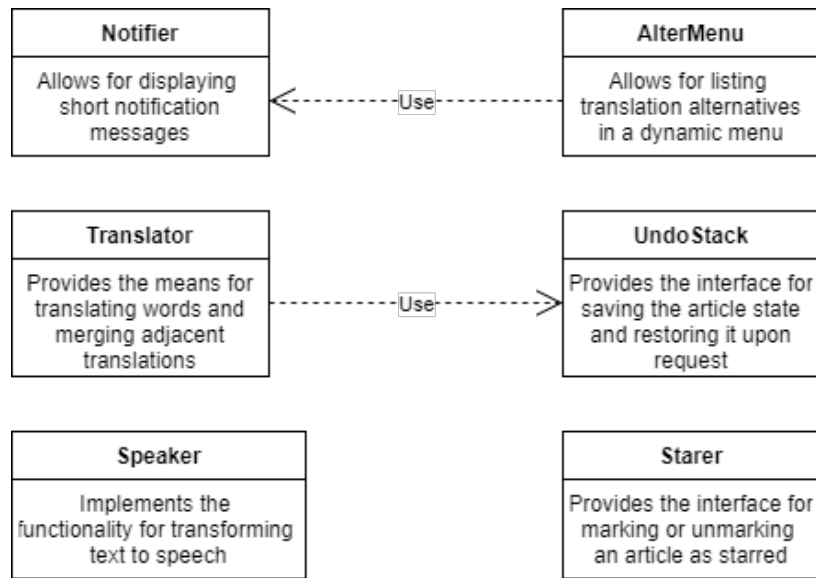
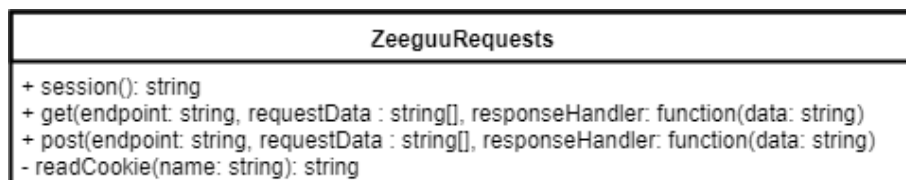


Figure 3: Front-end architecture overview

### 5.2.1 *ZeeguuRequests*

As we have previously underlined, the reading platform is built around the Zeeguu API [2]. The system interacts with this API in order to retrieve useful information, such as translations or alternatives to those. Also, it serves as an endpoint for communicating the user's interaction with the platform, such as suggesting translations, liking an article or starring it.

The Zeeguu API provides the means for communicating with it as a set of REST endpoints, usually in the form of a POST or GET request. To provide a consistent as well as abstracted manner to perform these requests, *ZeeguuRequests* class (Figure 4) is the one that implements and provides the GET and POST methods. These allow to specify the endpoint that will be contacted, the data to be sent and the eventual callback method when Zeeguu replies. Moreover, it automatically takes care of the authentication details when contacting the API, as it retrieves the appropriate *session ID* parameter from the cookie, where it is stored after logging in on the Zeeguu platform.

Figure 4: The static *ZeeguuRequests* class

Given that all the requests are performed asynchronously and the responses are received with unpredictable delays, the need for callback methods arose in order to allow the application to remain usable and operable at all times.

### 5.2.2 *Translator*

The ability to translate is one of the highlight features of the reading platform. Its functionality was carefully crafted to allow for seamless interaction.

The translator has three main functionalities (Figure 5):

1. waiting for translation requests upon which a merging pre-processing function is used
2. contacting Zeeguu to retrieve the translations
3. inserting the received translations.

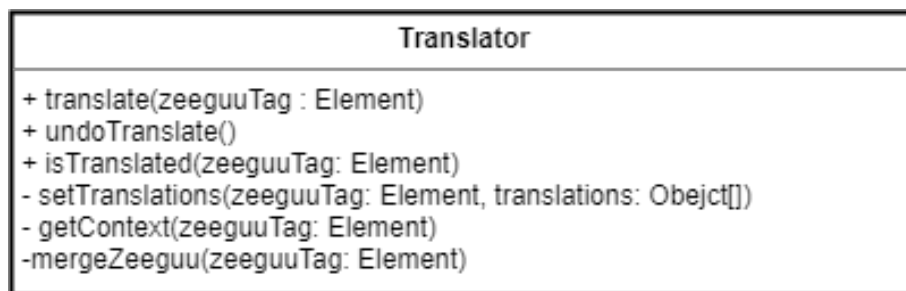


Figure 5: Translator class

The system was designed to only support translating one word at a time. However, the functionality added along the way is the ability to merge consecutively translated words into a 'translation bubble', providing the user with expectantly an even better translation for that sequence of words.

We have underlined, that in the back-end, words are separated and surrounded with the custom 'zeeguu' tags. This serves two significant purposes:

1. The ability to accurately and easily determine the word that was 'clicked' by attaching custom listeners to these custom tags.
2. Facilitating merging of consecutively translated words, that essentially boils down to identifying consecutive 'zeeguu' tags, for which a translation has already been retrieved and then merging that sequence under a new 'zeeguu' tag, for which the translation is then requested.

Because translations are not instant, a subtle pulse animation is attached to the in-process 'translation bubbles'. This provides appropriate visual feedback to the user to underline on ongoing activity, but it also does not distract

the attention if the person reading wishes to temporarily move on with their reading.

A translation is always appended with a CSS after rule <sup>2</sup>. When the translation arrives, it is inserted and the surrounding paragraph is automatically adjusted to accommodate the extra text. This however means that if there are any perceivable delays in the translation process, then the paragraph the user is currently exploring, will suffer from textual adjustments when the translations are inserted into place. This movement is of course distracting. Our attempt to minimize this as much as possible is to anticipate from the beginning the length of the upcoming translated word. The simplest rule is to use the length of the original word and replace its characters by something neutral. We have discovered that replacing each character with two dots gives the best results, both in terms of being unobtrusive and doing a decent job at minimizing any additionally needed textual adjustments.

### 5.2.3 *Speaker*

The text to speech functionality uses the `SpeechSynthesisUtterance` interface of the Web Speech API to complete speech requests <sup>3</sup>. The advantage and simplicity in using it is that it suffices to supply the language and the text to be spoken. It can of course be further tweaked by setting the pitch or the tempo for the voice, but this extends beyond our needs, at least for the current state of the platform. The class is thus rather minimalistic and its diagram can be seen in figure 6.

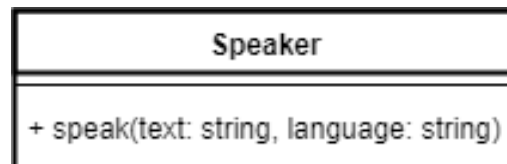


Figure 6: Speaker class

### 5.2.4 *AlterMenu*

The need to show alternative translations in a compact and convenient way is solved by our solution that lies in the `AlterMenu` class. It uses a dynamically manipulated `html div` container that gets constructed, placed and then opened right under the word for which alternatives have been requested to be displayed. The translations are saved as attributes of the custom `'tran'` tag, thus, easily retrieved and inserted into the `AlterMenu` as separate rows (buttons). Last

<sup>2</sup> <https://developer.mozilla.org/en-US/docs/Web/CSS/::after>

<sup>3</sup> <https://developer.mozilla.org/ro/docs/Web/API/SpeechSynthesisUtterance>

but not least, there is a field which allows to manually input alternatives. The overview of this class is offered in the figure below:

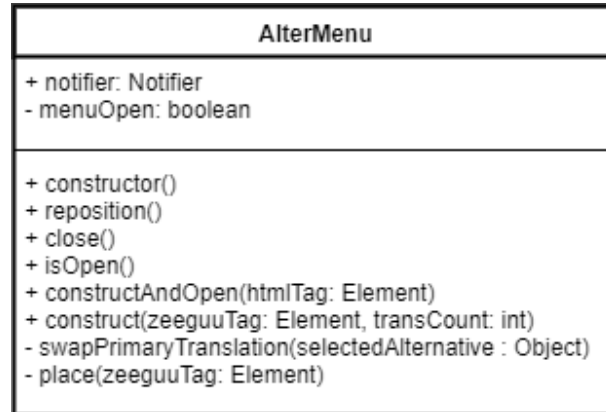


Figure 7: AlterMenu class

The AlterMenu will swap the translation shown in the ‘translation bubble’ with the one chosen from its list. If however no alternatives are available a Notifier object is used to inform the user about this situation. The user can still contribute with a translation if he or she so desires.

In the event that the user chooses an alternative or provides one, Zeeguu gets notified of the action. This allows for the creation of a small feedback loop that eventually aims to improve the translations delivered by the Zeeguu API.

### 5.2.5 Notifier

In order to inform the user about certain events in the system, we have built the Notifier class that underwater uses the MDL `snackbar` component. The need to wrap it in a custom class arises from the fact that this snackbar will be displaying messages in a queue manner, even if these messages are repeating themselves. Such behaviour is obviously unwanted.

The Notifier class (Figure 8) takes care of this by preserving the information about the text of the current notification. If a request with the same text as the one currently in view comes in, it is simply ignored. Everything is thus nicely abstracted. The Notifier behaves almost like a static class, thus it does not need to be instantiated as a separate object when used. To use this notifier, the method `notify()` with the message as a parameter is sufficient.

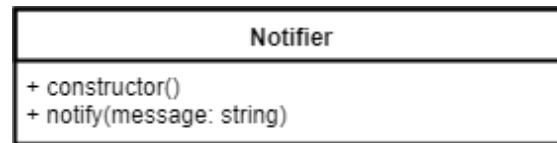


Figure 8: The Notifier class

### 5.2.6 *UndoStack*

Due to the translation functionality being so accessible, a user might erroneously translate a word. It therefore, sometimes occurs that undoing a translation might be needed. Since we are manipulating the html of the page on each translation, the easiest way to add undo functionality is to preserve the state of the article content in html format before each translation request, onto a stack.

Once an undo request comes in, the state from the top of the stack is popped and replaced with the currently displayed one. When the stack is empty, the undo request will be ignored. An overview of this class is presented in the figure below.

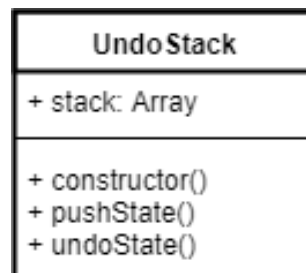


Figure 9: UndoStack class

The advantage of using this solution is the ease in its implementation. However, it depends on the article template and it can also be memory wasteful.

### 5.2.7 *Starer*

Zeeguu provides the ability to mark specific articles as 'starred' and then retrieve those in a list. This gives the user the possibility to save an article for a later reading session since it is easily identified under the starred section of the reader (more on this in the Prometheus thesis).

The implemented functionality lies in the Starer class (Figure 10). It gets the reference to the 'star' component on the page and then toggles its state accordingly. Moreover, this state is synchronized with the Zeeguu API, using the ZeeguuRequests class. Visually, the javascript code toggles a class attribute

that as a result swaps two icons depending on whether the article is starred or not.

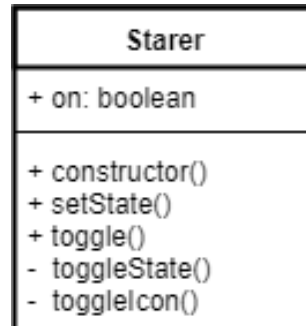


Figure 10: Starer class

#### 5.2.8 *UserActivityLogger*

The interaction of the user with the system is logged using the `UserActivityLogger` class (Figure 11). It allows us to later analyze the behaviour and then eventually decide on valuable improvements for the system. The logs are sent to a remote endpoint and since the functionality of the logger is implemented by using the `ZeeguuRequests`, it handles everything asynchronously.

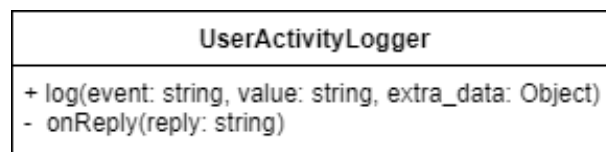


Figure 11: `UserActivityLogger` class

The method `log()` of the `UserActivityLogger` keeps things minimal and properly structured. As argument it requires a string that defines the type of event that is sent. Additionally, a logging event can also be accompanied by some 'value' and even 'extra-data'.

Additional information about the dependencies of this project can be explored in the appendix B.

## RESULTS AND EVALUATION

---

In order to evaluate the engagement of our users with the product, we conducted several analysis procedures from which we gathered usage data. This includes feature interaction, content interaction and platform acceptance as a whole. We tried to understand and evaluate whether the reading platform does indeed provide the users with a consistent and useful environment to digest their favorite content in the language(s) they study. We wanted to observe whether the things people read are indeed personalized and even though there might be trending topics, everybody tends to read the subjects that interests them. The above are questions in relation to **RQ2**. Moreover, we want to estimate whether the platform is simple enough to be seamlessly used and thus determine whether the tools that are provided to the person reading are actually used and whether the addition of a 'like' or 'star' button can help in determining parameters that define user interest. The latter questions are in relation to **RQ1**. Last but not least, the gathered insights have been extracted from anonymized data in order to preserve users' confidentiality.

The obtained statistics are retrieved with the use of either Hotjar (see appendix B) - an analytics and feedback framework or from the logged events to the Zeeguu back-end.

### 6.1 USER SATISFACTION

The best way to understand how satisfied the users are is to talk directly to them. To automate and speedup this process, we have sent out polls through Hotjar, in which users could answer our questions and rate the system itself. Additionally, we asked them for feature ideas that would improve their experience and we inquired about potential inconsistencies or problems with the platform, to determine the robustness of the system. To give an idea of what these polls look like, the table below gives a glimpse of the feedback we have received.



Device	Browser	OS	Quick question: How can we improve your reading experience?
desktop	Chrome 59.0.3071	Chrome OS 9460.60.0	The apostrophe in a translation is becoming a code (&#39;)
desktop	Chrome 58.0.3029	Mac OS X 10.12.5	hmm. let me read first :)
desktop	Chrome 57.0.2987	Windows 10	I clicked on the word "wakkert" in an article an zeeguu could not give me the translation. It seems that it has difficulty to recognize verbs with a separable particle. In this case, the verb was aanwakkeren.
desktop	Edge 14.14393	Windows 10	I think it is a good idea if you can add a stop watch with the article reading.
phone	Chrome Mobile 58.0.	Android 7.0	De vertaling moet met de van dale woordenboek in Nederlands worden vooral de hoger niveau.
desktop	Chrome 58.0.3029	Windows 10	Maybe add some decoration so that it does not look like the entire page is filled with text
desktop	Chrome 58.0.3029	Windows 10	If you place the images form the articles I would understand the context better.
desktop	Edge 14.14393	Windows 10	When you select a word in a text, sometimes it can be useful to remove the word because it wasn't your intention to click.
desktop	Chrome 58.0.3029	Windows 10	There are English texts in my list with French texts

Figure 1: Feedback poll example

Some of the things that we have learned from this feedback are:

1. Internet Explorer does not work - Even though in initial stages of development, IE was a supported browser, we were forced to drop it due to it being legacy software, not supporting modern day functionalities required for the system.
2. Ability to save articles to a separate list - This is what pushed us into bringing the starring feature to the system, which is integrated with the related Article Subscription system [10].
3. Preserve some visuals - People think their reading experience would be greatly improved if some of the pictures in the articles were still present. Thus, this point has been added to the future work section.

Besides issues and/or improvements, we have received feedback that serves as appropriate validation for the perspective that the reader is a good replacement for the traditional textbook material (Figure 2 and Figure 3).








	OS	DATE	Do you prefer this reading platform (not considering the exercise platform) to y ...
		5th July	The textbook materials are repetitive and boring
		3rd July	I can choose the topic of the texts I read
		2nd July	Yes, it's easier to find translations of difficult words. Also, you can choose what article you want to read.

Figure 2: Preference for the reading platform poll

If you wanted to read something in the language you study, what would you reach out for first?

12 responses

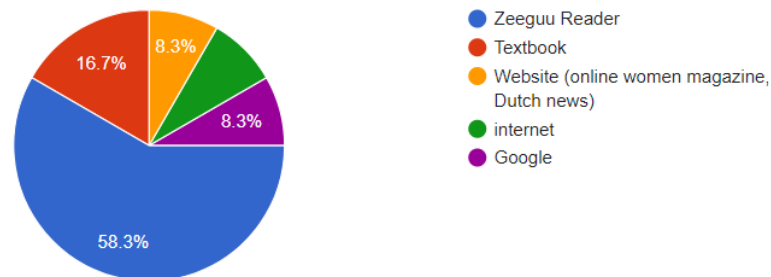


Figure 3: Preference for the reading platform chart

## 6.2 HEAT MAPS

Another way to analyze a platform is by looking at heat maps of its usage. These allow us to understand whether users are having troubles interacting with the UI and how intensively are certain elements on the platform used. To give an example of what the usual user interaction on a website is, we present the figure below (source <sup>1</sup>):

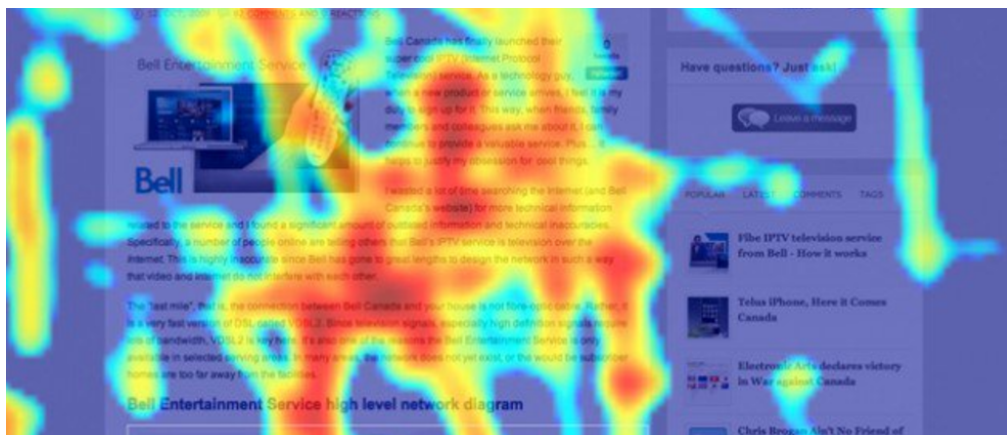


Figure 4: Heat map example

<sup>1</sup> <https://conversionxl.com/19-things-we-can-learn-from-numerous-heatmap-tests/>

What we notice is that users' activity is rather chaotic as they are exploring some textual content, going over pictures and other alternative links, hence these are possible distractions from the reading experience.

Hot jar is again the tool that helps us in tracking and generating heat maps for our system (Figure 5). A more uniform interaction can already be observed on our reading platform.

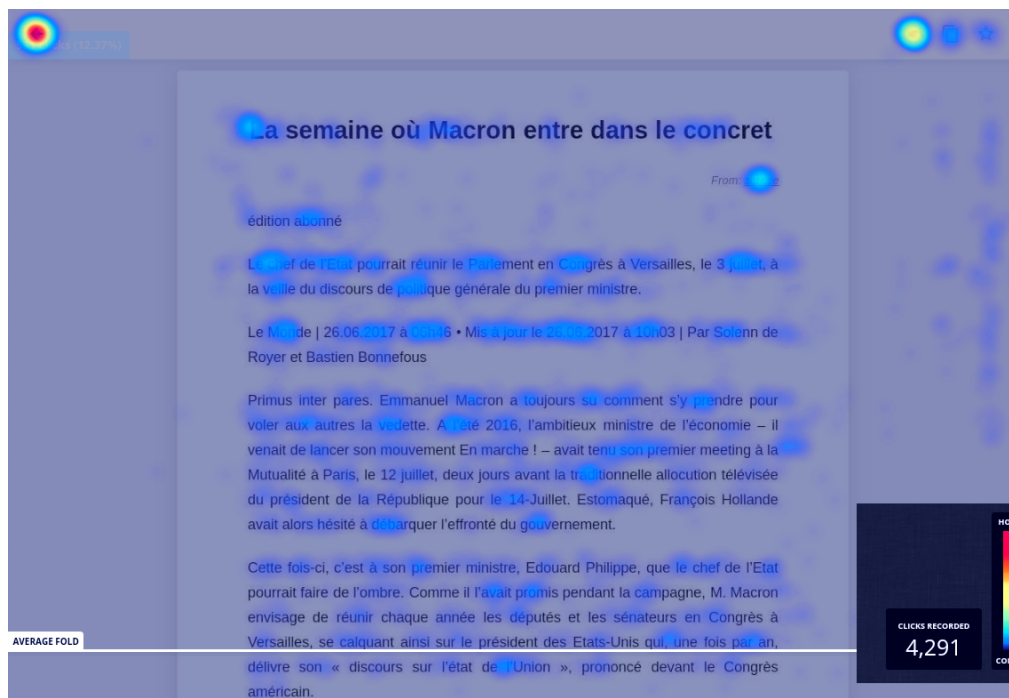


Figure 5: Zeeguu Reader heat map example

This heat map is based on a sample from the total number of registered clicks (4291). The insights we gather from it can be summarized to the following:

1. The back button (top left corner) appears to be heavily used, pointing towards the fact that the two subsystems of the multilingual reader: Prometheus and Apollo, integrate together properly, encouraging users to go back and find other things to read, instead of simply leaving the platform only after exploring a single article.
2. There appears to be an even distribution of clicks within the text area, which suggests that the translation feature is properly and actively used within articles.
3. In the top bar, on the right side, we see a decreasing tendency in the use of the: undo, copy and star button respectively. This aspect however, needs to be treated a bit differently, as the star button is not used on every article

and even if it is, then it is usually clicked at most once per article. On the other side, the undo button appears to serve its purpose very well, as it is actively used (more on this in the next section and in Figure 6). The copy toggle however, shares a much smaller user click interest, guiding towards the idea that it may be a feature with too small of an impact to be preserved. This however would be efficiently checked by performing some A/B testing in future development phases.

Even if these heat maps provide good knowledge about the usage of the platform, the static nature of them does not allow us to analyze some of the dynamic components in our system. The following section tackles this problem from another angle.

### 6.3 STATISTICAL ANALYSIS

What concerns the intensity with which the platform's components are used, we derive statistical analysis based on the events that have been logged to the Zeeguu back-end. We are presenting the barplot in Figure 6 from which several observations are deduced.

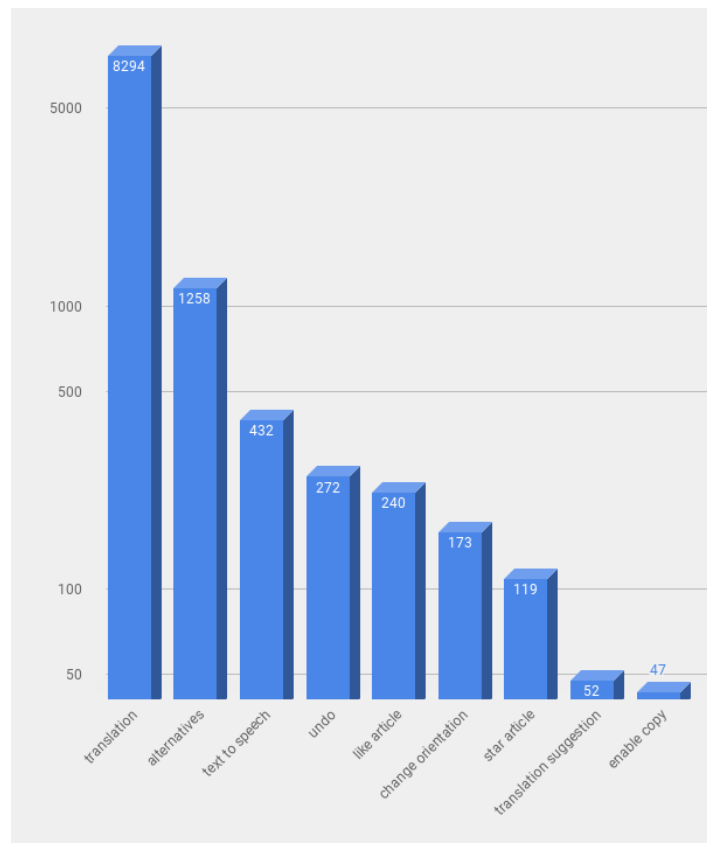


Figure 6: Barplot of the components' usage in number of requests (log scale)

We can clearly see that the main feature of the system - translations - is indeed, as expected, the most used one, guiding us into thinking that it definitely brings value to the reader.

Second to it is the feature that offers to the user translation alternatives. This means that people are eager to explore all the meanings for a translation, they do not take the default version for granted, but it also might indicate that in about 15.1% of the times, the primary translation choice does not fully satisfy the person reading. Nevertheless, they usually are able to find appropriate translations from the alternative ones, since the feature for sending translation suggestions appears to be used very lightly.

Taking a look at the number of users that interact and use the features of the system, we have computed the numbers in the figure below:

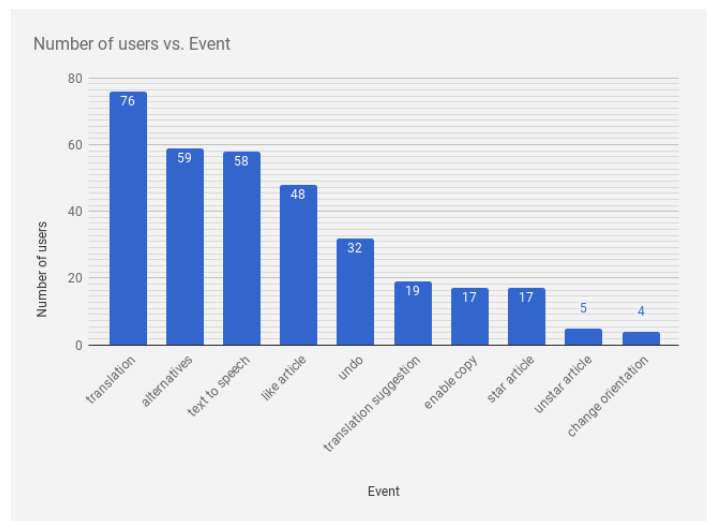


Figure 7: Barplot of the components' usage in number of distinct users

It is easily noticeable that this plot follows a similar trend to the one in Figure 6. In the top five most used features, we also identify:

1. text-to-speech
2. undoing a translation
3. liking an article

About 5% of the times, if a translation was retrieved, a pronunciation for it was requested as well. We also looked at the number of times the same word or phrase was pronounced by the same user. This data ranges from one single pronunciation to 14 pronunciations for the same word (phrase). The size of this interval is mostly due to the users' different proficiency in a certain language and the difficulty in pronunciation of the word (phrase) itself. Nevertheless, on average, the number approaches 1.66 pronunciation requests for the same

piece of text, suggesting that users are generally sufficiently content with a pronunciation after hearing it the first time.

What concerns the undo functionality, we have calculated that on average a user performs 4.9 undo requests per article.

One of the conclusions that can be drawn from the above is that the things most related to the translation functionality are among the highlights of the features of the system.

Another interesting point regarding the way the users interact with the platform can be deduced from the following table:

avg. # articles explored/user	avg. # total translations/user	avg. # translations/user/article
8.44	114.47	16.7

Figure 8: Average statistics

The data in this analysis has been collected from about 70 distinct users from a period of around 2 weeks. Respectively, we see that on average, each user has explored about 8 articles, requesting about 16 translations per article. What we can infer from this is that the ability to translate words within articles serves as a good and experience enhancing functionality, while still allowing to maintain the focus on the reading experience per se.

Looking at this data from another perspective, we built the following histogram (Figure 9):

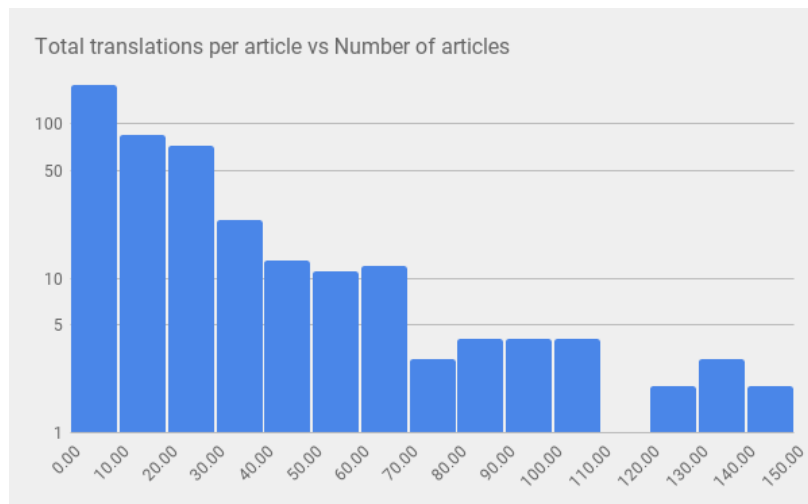


Figure 9: Histogram of the total translations per article vs Number of articles (log scale)

From it, we observe that most of the articles (about 300 at the time of the creation of this plot) that have been explored, have received up to 40 translation requests from all the involved readers. This suggests that users are reading

content of an appropriate level of difficulty for them. Thus the users tend to use the translation capabilities of the system, only when really needed, which is, as mentioned before, the actual intended behaviour of a good extensive reading experience. There are also a few outlier articles, where the number of total translations exceeds 150 translations, but those were left out of the histogram to better accentuate the meaningful data.

Last but not least, we want to understand whether the language learners are benefiting from having the advantage of choosing the content they have interest in. To support this claim, we present the scatter plot in Figure 10.

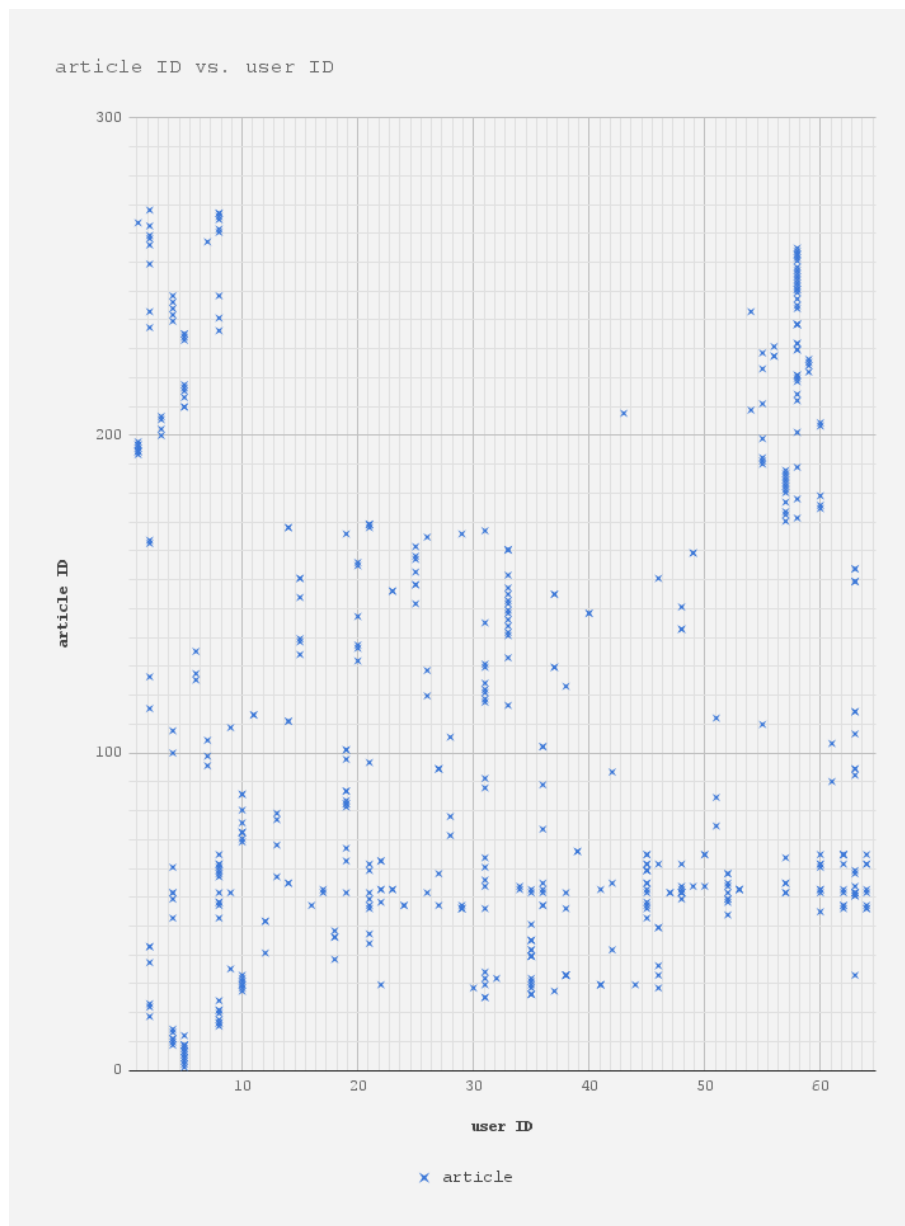


Figure 10: Article exploration vs users

This plot helps us understand one very important thing: from the almost 300 different articles that were explored by the users, there does not seem to be any universal choice for a specific article. This can be inferred from the fact that there are no horizontal line patterns in the scatter plot, meaning that even if there are articles read by more than one user, overall, each of them had his own, personalized set of reading content.

Besides having personalized content, the system also tries to achieve multi-platform support. To back this up, in figure 11 we provide a chart that presents the usage across many different browsers.

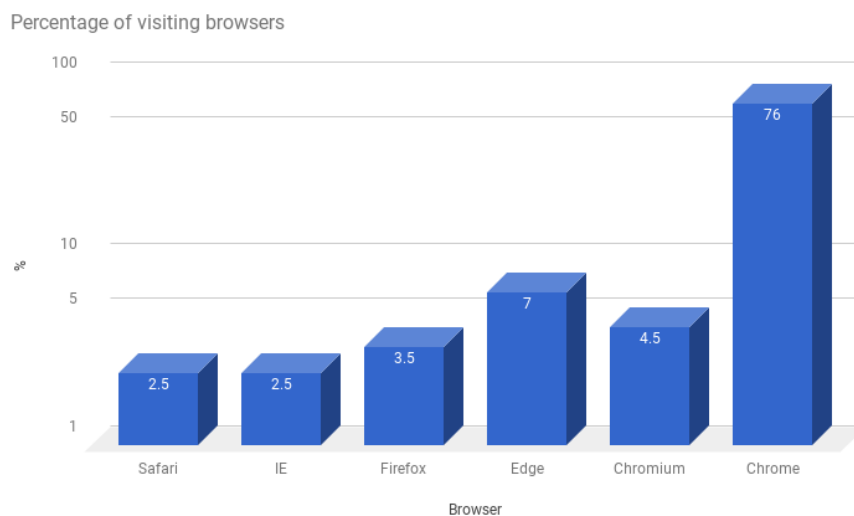


Figure 11: Different browser usage (log scale)

The highest tendency gravitates towards Google Chrome Browser, which is to be expected given the today market trends. Moreover, the fact that we have dropped support for Internet Explorer, affects only about 2.5% of our users. Not only that, but since Internet Explorer is slowly being faded out and replaced by Edge, the issue for IE support is practically negligible.



## 6.4 DISCUSSION

As a result from the research and the conclusions we have drawn in the previous sections, several perspectives with respect to our research questions can be formulated:

1. The platform successfully allows the users to enjoy their personalized reading content. They are engaged and make good use of almost all of the tools that the platform provides them with. As we have seen and confirmed, people use the opportunity to read only the content that they find interesting. To add to this however, a further level of research needs to be applied in the future, to determine whether once a user has started to explore an article, is it actually interesting for him or her, or do they abandon it midway. We tried to compensate for that by providing them with the ability to 'like' an article, which then has a stronger indication that indeed the content they have identified from the Prometheus platform, is actually enjoyed in the Apollo part.
2. As the focus was on creating a streamlined and lightweight system, the users have reported high satisfaction incentives. Not only that, but we observed the system performing properly on all of the platforms of development (i.e. browsers). Ultimately, more testing needs to be performed to determine even more accurately the usability of certain features and eventually to identify the need for novel ones.
3. We understood that the feedback system, even though useful and easy to use, can be extended and improved. We would like to have more leverage into how and what we analyze from the user behaviour and interaction with the system.

The research questions have thus been answered successfully. Nevertheless, we believe there is potential for more analysis with respect to them as the platform will grow and evolve.

## CONCLUSION

---

Even if the project had a predefined outline, we tried to remain flexible along the entire development process. Due to having a solid and well established workflow (see appendix A) we have been able to accommodate additions and changes to the project in a iterative manner, continuously improving on their quality.

The system itself was designed with maintainability in mind. Developing a web platform proved to be a rewarding decision, not only for the reduced complexity in the development process, but also for the high malleability of the product towards change. Additional requirements were usually easily accommodated into the architecture, while testing and bug fixing was most of the time straightforward.

The evaluation of the platform indicates that users are indeed actively engaged with the content that they interact with and that the minimal set of tools provided to them in the reading environment, proves to be helpful and useful. We were able to gather valuable insights from the user satisfaction metrics and surveys that were sent out.

The end product meets the initial requirements and scores high in the user satisfaction department. Consecutively, it sets a good and strong basis for further improvements.

## FUTURE WORK

---

An important aspect that is bound to this project is the ability and intention to improve it and evolve it as times goes by. The project shows a lot of potential and the only way to nourish that potential is by paying attention to the future work.

Several aspects of the platform have been identified to be good candidates for further feature definitions, most of which are supported by the user feedback we have received.

### 8.1 DICTIONARY SUPPORT

Sometimes, simply translating a word might not be sufficient to fully and properly understand its conveyed meaning. There are a lot of times, when the translation for a word is not completely different from the original one, yet, the fact that even in the user's native language (in this case the language we translate to) that word might be unknown, underlines the importance of dictionary functionality for this system.

### 8.2 PRESERVE SOME VISUALS

People have pointed out that numerous articles have at least one image that considerably enhances their grasp of the content. Currently, being presented solely with a page of text, if the article is longer than usual, it tends to transmit an initial impression of intimidation. To stimulate user engagement even more, we might look into preserving some of the original visuals of the articles.

### 8.3 PRESERVE ARTICLE STATE

Going back to the idea of articles that are longer than usual, it would be rather useful to save the progress on an article and then later, allow the user to seamlessly continue from where he or she left off. Right now there is the

possibility to star an article and have it saved to a custom list, but after returning to it, all the previously retrieved translations are lost.

#### 8.4 PARTICLE VERBS SUPPORT

Currently, the system translates single words, or groups of words consecutive to each other. However, things such as particle verbs are not yet supported. Particle verbs are word constructions where the verb is accompanied by a particle, but these do not necessarily come together. Depending on the language, they get separated by one or more words, but they convey the correct meaning only when translated together. Interface support for this would be welcomed.

#### 8.5 READING MODES

Reading in darker environments while having a white background causes a lot of strain on the eyes. The idea is to offer the possibility to let the user choose to adapt this to suit his or her preferences. Either having the ability to invert the theme and read light gray text on black background or change the complete white background to something more toned down, like sepia, which is known to be more pleasing to the eye. This will eventually help the user achieve comfort no matter when and where he or she prefers to read.

#### 8.6 READING RATE

Another important aspect that determines the user's ability to explore and grasp the content of an article is his or her reading speed. It is one of the parameters that can be estimated by determining the difference in time between the event that marks a user accessing an article and then the time when the article was left, together with the amount of words that were read.



# WORKFLOW

---

Nowadays, the need for continuous improvement, especially in the Software Engineering field, is a well established aim. This implies that a system should be and will be - subject to changes, additions, reductions and who knows what else. Consecutively, as a software designer and developer, we have to always think with regards to that idea and act accordingly. This philosophy is nicely summarized yet greatly encompassed by the idea of Scrum - an iterative and incremental, agile software development framework. An idea parented and developed by Jeff Sutherland in his popular book on the same topic [9].

The concept of Scrum and its backing framework represent extensive tools and principles to be applied in a development process. To be able to cover and adapt entirely to this working methodology requires a rather advanced level of expertise. We however, tried to approach it in a more relaxed manner, but still, successfully following its core focus. To outline the workflow that we have used, we can underline the following points:

1. The development process was divided into weekly iterations.
2. Each iteration was bound to a Sprint specification.
3. A sprint consisted of a number of issues to be addressed during its duration, usually decided upon with the involvement of the supervisor - Mircea Lungu.
4. During a sprint, we would have daily meetings for discussion, analysis, design and development.
5. The development process was co-reviewed with L.A.H. van den Brand, by sending and reviewing each other's pull requests.
6. A sprint would generally end with a pull request from a release branch, where the main supervisor would be responsible for the review.
7. Weekly meetings with the customer/supervisor would help us determine the next milestones for the upcoming sprints.

## DEPENDENCIES AND TOOLS

---

We will limit ourselves to only presenting a few of the tools and dependencies of the system. The remaining components are thoroughly documented in our online documentation, present on Github pages<sup>1</sup>.

### B.1 FLASK

Flask <sup>2</sup> is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It allows to run a RESTful Python Server that is simple to use, comprehend and maintain. Furthermore, it allows for easy sharing and code reuse between Zeeguu's services since these also implement their functionality using the same microframework. Nowadays, Flask finds itself to be the most popular Python web development framework on GitHub.

### B.2 NEWSPAPER

Newspaper [6] is an amazing python library for extracting and curating articles. It is extremely straightforward in its use and has a lot of embedded functionality. It allows to download the page of an article, parse the page to retrieve the content of the article and even allows to perform NLP analysis on the article to identify keywords in the text, topics, etc.

### B.3 HOTJAR

Hotjar <sup>3</sup> is a powerful tool that reveals the online behaviour of the users, by providing with both feedback and analysis tools. The former is tied to sending polls and surveys to users while the later is concerned with building heat maps of the website usage, as well as recording the user activity itself. The simplicity in integrating it into the system is what makes it a very viable option.

---

<sup>1</sup> <https://mircealungu.github.io/Unified-Multilanguage-Reader/>

<sup>2</sup> <http://flask.pocoo.org/>

<sup>3</sup> <https://www.hotjar.com/>

# BIBLIOGRAPHY

---

- [1] Timothy Bell. Extensive reading: Why? and how? *The Internet TESL Journal*, 4(12):1–6, 1999.
- [2] Mircea F. Lungu. Bootstrapping an ubiquitous monitoring ecosystem for accelerating vocabulary acquisition. In *Proceedings of the 10th European Conference on Software Architecture Workshops, ECSAW '16*, pages 28:1–28:4, New York, NY, USA, 2016. ACM.
- [3] C. McCarthy. Reading Theory as a Microcosm of the Four Skills. *The Internet TESL Journal*, 5(5), 1999.
- [4] S. Namhee. The Effects of Extensive Reading on Reading Comprehension, Reading Rate, and Vocabulary Acquisition. *Reading Research Quarterly*, 52(1):73–89, 2016.
- [5] J. Oosterhof. Making reading in a second language more enjoyable. 2016.
- [6] Lucas Ou-Yang. *Newspaper - Article scraping and curation*.
- [7] Willy A. Renandya. The power of extensive reading. *RELC Journal*, 38(2):133–149, 2007.
- [8] L. Schwab. Using rss Feeds to Support Second Language Acquisition. 2016.
- [9] Jeff Sutherland. *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business, New York, NY, USA, 1st edition, 2014.
- [10] L.A.H. van den Brand. Prometheus: Efficiency and Usability in a Personalized Multilingual Feed Manager. 2017.