Web accessibility
*A middleware prototype for visually impaired users*
Abstract

In the context of a society where the Web is present in many aspects, there is a significant amount of visually impaired users whose experience is far from being satisfactory. There are technologies aiming this problem but still with no full success. The problem addressed in this project is the existing gap between the visually impaired users and the solutions being offered to them. As a solution to this problem, a middleware prototype is developed. It acts as a web application so the user does not need to install anything. The middleware also offers different adaptations to the user such as amplifying lens, text narrator, and others. The solution was tested by visually impaired users and it received an overall positive result. Some features like the amplifying lens received a good value and some of them need further improvement.

Keywords: web accessibility, visually impaired, prototype, middleware, assistive technologies, low vision, web adaptation
Preface

This project aims to propose a solution for the visually impaired users when navigating the World Wide Web in the form of a middleware prototype. The project helped me to learn in a technical and a social way how the visually impaired users relate and interact. It was a rich learning process which showed me another aspect of web design and development and also a great insight of user interfaces.

I would like to thank my project coordinators Doctor Arianit Kurti and Bahtijar Vogel for the technical guidance and feedback. The Doctor Mexhid Ferati for his Human Computer Interaction for visually impaired people advices. The professor of UPC – FOOT Eulalia Sánchez for her scientific, optical and optometry approach and counselling. And finally, to the main target of the project, the visually impaired users that collaborated in the validation test.
## Contents

1 Introduction  
1.1 Project overview ........................................... 1  
1.2 Motivation of the author .................................... 1  
1.3 Purpose and research questions .............................. 2  
1.4 Scope and limitations ....................................... 2  
1.5 Target group .................................................. 3  
1.6 Outline ....................................................... 3  

2 Theoretical background  
2.1 Introduction ................................................... 4  
2.2 Visual impairments ........................................... 4  
2.3 Accessibility Guidelines Overview ......................... 5  
  2.3.1 WCAG 1.0 ................................................ 5  
  2.3.2 WCAG 2.0 ................................................ 5  
2.4 Current state of accessibility ............................... 6  
  2.4.1 Websites and WCAG 2.0 ................................. 6  
  2.4.2 Disabled users and WCAG 2.0 ......................... 7  
  2.4.3 Users and assistive technologies ....................... 8  
  2.4.4 Web designers and accessibility guidelines .......... 8  
2.5 Challenges arisen .......................................... 9  

3 Research approach  
3.1 Research overview .......................................... 11  
  3.1.1 Research motivation ..................................... 11  
  3.1.2 Research solution ....................................... 11  
  3.1.3 Solution reasoning ..................................... 12  
3.2 Methodology ................................................. 12  

4 Prototyping  
4.1 Requirements analysis ...................................... 14  
  4.1.1 Functional requirements ................................ 14  
  4.1.2 Non-functional requirements ........................... 15  
4.2 Design ....................................................... 15  
  4.2.1 System architecture .................................... 16  
    4.2.1.1 Model ................................................ 17  
    4.2.1.2 View ................................................ 17  
    4.2.1.3 Controller ......................................... 17  
  4.2.2 Use case ................................................ 18  
  4.2.3 Functional requirements design ....................... 18  
  4.2.4 Non-functional requirements design .................. 27  
4.3 Implementation ............................................. 28  
  4.3.1 Framework .............................................. 28  
  4.3.2 Released version ....................................... 28  
  4.3.3 Examples .............................................. 29
5 Validation

5.1 Usability test ................................................. 32
5.1.1 Conditions ............................................. 32
5.1.2 Usability survey ....................................... 32
5.2 Test results ................................................. 33
5.2.1 Participants ............................................ 33
5.2.2 Task success ........................................... 34
5.2.3 Middleware features ................................. 34
5.2.4 Middleware overview .............................. 36

6 Discussion ...................................................... 38

6.1 Problem solving ......................................... 38
6.2 Method reflection ........................................ 39

7 Conclusion ...................................................... 40

7.1 Conclusions ................................................. 40
7.2 Further research .......................................... 40
1 Introduction

Nowadays we use the Internet and websites for multiple types of tasks, from sending an email to our friend, check the bus timetable or the result of a sports game.

Since the uptake of the so-called Web 2.0, term popularized on late 2004 [1], new technologies and techniques for web design and construction have appeared. This new approach allows a great interaction between the web site and the user, switching the user from just a content consumer to a new interactive role.

However, this changes have not benefited all the users by the same weight. As the World Health Organization estimates, in figures dating 2013 \(^1\), in Europe there are 26 million people with visual impairments and 4 million people that suffer from sight-loss.

The evolution of the Web has not been paired with the evolution of accessible technologies for visually impaired users [2]. Some approaches have been done to improve this issue like screen readers, Braille output devices, or web design guidelines. However, those means did not fully solve the problem when being released nor have solve it now. Some of them because of not including all the problematic aspects and some of them because have not been updated over the time. To bring up an example, the last revision of one of the most used web design accessibility guidelines dates December 2008 [3].

A change in the overview of website design and a stronger inclusion of accessibility is needed given that visually impaired users still cannot fully benefit from the advantages this new Web 2.0 has been offering the last 10 years.

This projects aims to contribute to this issue offering a middleware prototype to improve visually impaired users’ web experience.

1.1 Project overview

This project is focused on studying the relationship between the visually impaired users and the web sites they navigate. There is a theoretical part which aims to research the current state of website design guidelines in terms of accessibility and how the involved parts interacts with them. This involved parts include a wide range of actors, from the disabled users to web designers. When finished the theoretical part some conclusive points are extracted. These points are used as starting lines for the practical part.

The aim of the practical part is to create a prototype of middleware that enables visually impaired users to have a better experience when surfing the web. This means to make websites more accessible to a group of users which cannot fully experience it as the majority of Internet users does. Given a certain website, the prototype is able to apply some transformations and adapt it to user’s needs.

Finally, some prototype validation is done in order to check the real state of the solution and to receive new guidelines to improve it in possible future works. The participants of the validation process are visually impaired users.

1.2 Motivation of the author

The selection of this project was mainly driven by two motivations.

One motivation is raised by the fact that web accessibility is one step behind the current state of web technologies. During the last ten years, websites and their technology have evolved in a really fast way. However, this evolution has not been paired with web

\(^1\)http://www.who.int/mediacentre/factsheets/fs282/en/
accessibility. This project brings an opportunity to study web accessibility and propose new solutions to this issue.

Another motivation is related with following the World Wide Web standards as I see them: universal, accessible, and decentralized. This project brings an opportunity to work on the idea of making the Web an accessible and available site to all people.

1.3 Purpose and research questions

For this project inductive methodology will be used. This methodology consists on a bottom-up approach. This type of methodology is used when some knowledge is missing. First, given a certain environment or background, one or more research questions are formulated. Then, to achieve the answers to those questions an inquiry or investigation process is carried out.

The background of this project is the interaction between visually impaired users and websites. Two question and one sub question are formulated. The first question will be answered conducting a theoretical process. The second question and its sub question will be answered conducting a practical process. The first research question \((RQ1)\) is \textit{How is the interaction between visually impaired users handled today?}. The purpose of this is to fill a descriptive knowledge gap. Information related to the current situation of visually impaired users and their relationship with web sites will be extracted from different scientific sources.

The second research question \((RQ2)\) is \textit{How can the interaction between visually impaired users and websites be improved?}. The purpose of this is to fill a procedural knowledge. Given the theoretical background from \(RQ1\), an improvement to the interaction between visually impaired users and websites in form of a solution will be proposed. Once this solution is presented, a sub question raises. This sub question \((RQ2.1)\) is \textit{Can the proposed solution improve the experience of web pages to visually impaired users?}. To answer this, a usability study with visually impaired users will be conducted.

1.4 Scope and limitations

There is a scope defined for this project in the theoretical and the practical part. For both of them, this scope establishes limitations and focus.

In the theoretical part, a Systematic Literature Review will be performed. This involves an analysis of the current situation of the interaction between visually impaired users and websites. The analysis will be based on actual studies which date from the year 2004 to the year 2013. This limitation has been set due to the low frequency of studies related to visually impaired users and web sites, web design, and related in previous years\(^2\). The analysis aims to extract different improvements that can be done in the topic of visual impaired users and websites.

In the practical part a solution to the research questions will be presented and validated. The solution presented consists of a prototype of a middleware. The requirements of this prototype will be extracted from the previous theoretical section. Given that this project is inside a Bachelor Degree only a prototype of the middleware and not the final version of it will be presented. After developing the middleware prototype, four visually impaired users will be subject of a User Validation Test. Being only four visually impaired users available, the results of this validation shall be analysed and studied in its

\(^2\)Approximately 88% of publications that appear on ACM Digital Library using the search \textit{visually impaired users} are inside the time range of 2004 to 2013
own context and not extrapolated into a solution to a larger population.

1.5 Target group

This project has two target groups. The first is the visually impaired users. The second is the engineers audience that develop websites and web solutions.

The first target group, visually impaired users, is formed by two types of users that suffer from a visual dysfunction. According to the World Health Organization

3\footnote{http://www.who.int/mediacentre/factsheets/fs282/en/}, there are four levels of visual function; normal vision, moderate visual impairment, severe visual impairment, and blindness. The collection of moderate visual impairment and severe visual impairment is referred as either low vision or visual impairment. The target group in which the theoretical background analysis focuses and the user target group for the middleware prototype is the low vision users.

The second target group is formed by all engineers that are involved in developing websites and creating web solutions. For this group the theoretical background study will contribute to give an overview of the current situation and what are the current gaps and challenges to complete. Also, the middleware will serve as a starting point and prototype of a possible solution for some challenges.

1.6 Outline

This report is divided in the following chapters:

- Theoretical background: Knowledge base built upon the analysis and study of scientific publications.
- Research approach: Overview of the solution approach and the methodology followed.
- Prototyping: Description and justification of the prototyping process.
- Validation: Presentation and analysis of the data obtained during the validation process.
- Discussion: Review of the knowledge gap filled.
- Conclusion: Overall review and future investigations lines.
2 Theoretical background

This chapter corresponds to the study and later analysis of the current state of the interaction relationship between the visually impaired users and websites. This section acts as a foundation for the practical part.

2.1 Introduction

This chapter founds the theoretical background of the project. This background corresponds to the analysis and subsequent interpretation of the selected data.

The chapter is divided into four sections:

- The first section defines what a visual impairment is and the different types of visual disabilities that exist. Not all visual disabilities are being taken into account in this project. This section establishes the target group.

- The second is an overview of the Web Content Accessibility Guidelines (WCAG). The selection of these guidelines instead of others is motivated by the fact that the authorship of them corresponds to the Word Wide Consortium (W3C), international community that develops open standards for the Web.

- The third is an analysis of the current state of web accessibility and its technologies and tools. This analysis consists of four subsections: websites and its compliance level to WCAG, visually impaired users and WCAG, visually impaired users and assistive technologies, and web designers and WCAG.

- The fourth section is a collection of the challenges arisen from the previous study. These challenges refer to the problems and gaps to be filled found in the previous section and related to the different areas analysed. For these challenges a brief description and justification is given.

2.2 Visual impairments

The target group of this project and the following reports and articles studied are the visually impaired people. The World Health Organization classifies visual function into four groups\(^4\). This classification is based on the best glass correction in the better eye:

- More than 0.3: No visual impairment
- From 0.3 to 0.1: Moderate visual impairment
- From 0.1 to 0.05: Severe visual impairment
- Less than 0.05: Blindness

The collection of moderate visual impairment and severe visual impairment is referred as low vision.

\(^4\)http://apps.who.int/classifications/icd10/browse/2010/en#/H54
2.3 Accessibility Guidelines Overview

Taking as main reference the World Wide Web Consortium (W3C), which is the organization who manages the standards to follow in the World Wide Web, this study follows the guidelines and techniques suggested by the Web Accessibility Initiative (WAI).

WAI has published several design guidelines and standards related to Internet and Web accessibility; Web Content Accessibility Guidelines (WCAG) which addresses accessibility in a web site, Authoring Tool Accessibility Guidelines (ATAG) which addresses accessibility in software that creates web sites, User Agent Accessibility Guidelines (UAAG) which addresses web browsers, media players, and assistive technologies, Accessible Rich Internet Applications (ARIA) which defines a method to create dynamic web applications, Indie User Interface (Indie UI) which defines a method for user actions to be communicated to web applications, and Evaluation and Report Language (EARL) which is a language format to facilitate test results processing.

2.3.1 WCAG 1.0

Web Content Accessibility Content Guidelines (WCAG) addresses the information in a Web site, including text, images, forms and sounds. The first version (WCAG 1.0) was published on May 1999 as a W3C Recommendation [4]. The guidelines are grouped by three priority checkpoints. Priority one checkpoints aim aspects that would be absolute barriers for users with disabilities. Priority two checkpoints aim aspects that would be significant barriers for users with disabilities. Priority three checkpoints aim aspects that would provide additional accessibility support for users with disabilities.

2.3.2 WCAG 2.0

WCAG 2.0 is the following version of the Web Content Accessibility Content Guidelines. This version was published on December 2008 as a W3C Recommendation [5]. The update was made on the guidelines and also in the evaluation criteria. WCAG 2.0 has 12 guidelines organized in four principles:

- Perceivability: Page content must be able to be changed in a perceivable form for users with vision or hearing loss.
- Operable: Pages must give sufficient time to assistive technologies’ users to complete inputs, avoid flashing that can cause seizures, and be navigable by different means.
- Understandable: Page content and interface controls must be understandable to all users.
- Robustness: The XHTML code of the page must be robust enough so assistive technologies can interpret and render it in an accurate manner.

The guidelines grouped in the previous principles can be tested under a given success criteria. The previous three checkpoints criteria described in WCAG 1.0 was reconsidered in terms of compliance levels. The reason of this change was that even fulfilling level three checkpoints some aspects of a website were still uncovered and critical for some disabled users. With this change, WCAG 2.0 has three compliance levels; A, always required for a site to be accessible, AA and AAA as more stringent and higher criteria.
In October 2012, WCAG 2.0 was approved as an ISO International Standard (ISO/IEC 40500:2012). This approval benefits countries and organizations so they can now adopt ISO standards and accessibility standards in an easier way.

2.4 Current state of accessibility

In the following points an overview of different aspects of accessibility is presented. Firstly, websites and its compliance level with WCAG 2.0. Secondly, disabled users and their experience with WCAG 2.0. Thirdly, disabled users and their experience with assistive technologies. Fourthly, web designers and their relationship with web accessibility guidelines.

2.4.1 Websites and WCAG 2.0

V. L. Hanson and J. T. Richards automatically evaluated A-level checkpoints that can be tested without human supervision on several websites [2]. This study was made using a software that analysed the level-A checkpoints of WCAG 2.0. It is worth stating that the first intention was to evaluate also level-AA and level-AAA websites but those were rarely found and therefore not included in the study. The websites used to conduct the study can be grouped in (a) Governmental websites (231 items) and (b) Most visited websites according to the Alexa ranking system (952 items)\(^5\).

After checking the websites some results were extracted from the situation of each principle:

- Perceivable: For this principle two guidelines where checked: the alt tag and the labelled inputs. The alt tag gives a brief description of an image. The labelled inputs gives a brief description of the paired input. There is a decrease over the years in missing alt tags and an increase in the use of labelled inputs. Also there is a relative increase in empty alt tags. The first trend may suggest that there is an increase of applying accessibility guidelines when designing websites. The second trend may suggest that the usage of some accessibility tools is still not clear to designers. In this case, they are using the alt tag but without a valid value.

- Operable: For this principle two guidelines were checked: the skip navigation bookmark and headings. The skip navigation bookmark allows the user to skip to the main content. The headings gives a title to the content of the website. There is an increase in the use of skip navigation and headings. In the year 2012, 80% of government sites used skip navigation and the 100% used the headings. As the data shows, the usage of these guidelines is wide spread and well-used. Using headings also benefits a website to be better positioned in search engines and benefits the non-impaired user as well.

- Understandable: For this principle only one guideline was checked, the usage of the lang attribute. This attribute specifies the language of the website. In government sites there is an increase in the use of the lang attribute, also in unique top sites but with a lower increment. In the year 2012, 90% of the government sites were using lang attribute against a 58% of usage in the most visited websites.

\(^5\)http://www.alexa.com/topsites
• Robustness: For this principle only one guideline was checked, the number of parsing errors per page. These errors include elements that do not have start or end tags, are not properly nested, have duplicate attributes, or not unique IDs. There is a trend in decreasing the number of parsing errors/issues in both government and top sites. Most common web browsers tolerate this error and modern web design software assists the creator to avoid this errors. Despite the previous fact, 20% of the top sites and a 8% of the government sites tested had one or more of the described errors.

In general, the data shows that there is a score increase during the last 12 years in WCAG 2.0 compliance levels. However, the pages were only analysed at level-A checkpoints and this compliance level should be mandatory for all web pages as said by the W3C. Another point that needs to be stated is that not all A-level criteria was tested in the study indicating that some points need to be tested manually by humans.

The results may indicate that accessibility is being taking more seriously on websites and its implementation is being wide spread. However, as the study points out, some accessibility improvements may be a side-effect of good coding practices and the desire to be better situated in search engines.

2.4.2 Disabled users and WCAG 2.0

C. Power, A. Freire, H. Petrie, and D. Swallow [6] conducted a study involving visually impaired users evaluation on selected websites and their feedback when encountering accessibility problems. The study involved 32 users and a set of 72 websites were evaluated. Each user evaluated 16 randomly selected websites.

A first test was made on the websites regarding their compliance with WCAG 1.0 and WCAG 2.0. From the 16 websites only four (25%) were in compliance and of these four, two were level-A, one was level-AA, and one was level-AAA.

After the first test, the visually impaired user test was taken. In the second test the users had to measure how many problems they encountered in a website and qualify that problem in between the following range, going from the least important to the most: Cosmetic, Minor, Major and Catastrophic. The results of the test were the following; the mean number of problems encountered per website was 86. These 86 problems are distributed in an average of nine as cosmetic, 34 as minor, 27 as major and 15 as catastrophic issues.

After that analysis, the problems encountered by visually impaired testers were classified into three groups according to the website compliance to WCAG 2.0; (a) non-compliant, (b) level-A compliance, and (c) all levels of compliance. The results from the grouping was that 102 problems were from group (a), 73 were from group (b), and 67 were from group (c).

A final classification were grouping the problems by their coverage in WCAG 2.0 and their corresponding implementation on the website was made; the first group (a) was problems not covered, (b) corresponded to problems covered but not implemented, and (c) problems covered and implemented. The percentage distribution was the following: 49.6% to (a), 42% to (b), and 8.4% to (c) group.

Given the previous data, the cited study extracts that for WCAG 2.0 there was no significant decrease when comparing user problems found between level-A websites and non-compliance websites. Also, only half of the problems encountered by test users were covered by WCAG 2.0. Finally, the study implies that WCAG 2.0 has not had the ex-
pected effect regarding accessibility issues and that it is time to change the problem-based paradigm currently used and find new ways to solve the accessibility issue [6].

2.4.3 Users and assistive technologies

In this section the relationship between users and assistive technologies is discussed.

A. Brown, C. Jay, A. Q. Chen, and S. Harper conducted a study of the use of assistive technologies during the period 2006 to 2009 [7]. The most used technology was the screen reader. The most common choice of screen reader is the proprietary software JAWS\textsuperscript{6} and in second place, also a proprietary software, Windows Eyes\textsuperscript{7}. However, there is an increasing trend of using open source alternatives such as Orca\textsuperscript{8}, from the Linux desktop environment GNOME, and NVDA\textsuperscript{9}. The reason for this trend might be the increase in popularity of open source solutions and the high price of proprietary solutions. For example, a 2013 standard license of JAWS software is 895\$\textsuperscript{10}.

In Guidelines are only half of the story: Accessibility problems encountered by blind users on the web a study with the purpose of determining the user experience when viewing and interacting with dynamic websites using their choice of assistive technology was made. Ten dynamic web pages were written for the study and 13 testers, visually impaired users, were recruited by a volunteering post on BCAB mailing list (British Computer Association of the Blind). The participants were asked to give feedback about their reactions when viewing the websites. The results showed two important facts. The first is that all ten tested screen readers were able to cope with new, dynamically appearing content. The second is that only one out of ten screen readers tested notified the user that new dynamic content had appeared in the website.

Finally, the study highlights the difference between making information technically accessible and making it accessible in an efficient manner. This statement is based on the way that screen readers manage new information appearing in the websites. The technologies adapt and react well to new content but most of them fail in the process of communicating to the user about this new content. This failure does not allow the visually impaired users to fully experience and benefit from dynamic web pages that use web 2.0 technologies.

J. T. Richards and V. L. Hanson implemented a software to make user requested websites more accessible [8]. This software is based on a proxy that receives the HTTP answers and transforms the received websites to a more accessible format. A user study was conducted and the analysis showed an important point; the visually impaired users had difficulties during the installation process. Usually, the installation process requires different steps and some configuration. Since the assistive technology is not yet installed, the user does not have a good enough support to go through this process. This stage remains a challenge and makes visually impaired users avoid installing software by themselves.

2.4.4 Web designers and accessibility guidelines

This section evaluates the level of knowledge and awareness that the web masters, web designers and web developers have of both Governmental accessibility guidelines and

\textsuperscript{6}http://www.freedomscientific.com/JAWSHQ/JAWSHeadquarters01
\textsuperscript{7}http://www.gwmicro.com/Window-Eyes/
\textsuperscript{8}https://wiki.gnome.org/Projects/Orca
\textsuperscript{9}http://www.nvaccess.org/
\textsuperscript{10}http://sales.freedomscientific.com/Product/340014-001/JAWS_Home_Edition.aspx
The method used to determine this levels was a survey conducted by J. Lazar, A. Dudley-Sponaugle, and K.-D. Greenidge [9]. The survey involved nine closed-ended questions, three open-ended questions and three mixed questions. The survey was posted on the web and 175 participants took part in the study. Those participants were web masters, web designers, and web developers.

For the close-ended questions part, the study showed that most of the respondents (64%) are familiar with WCAG 1.0 and WCAG 2.0. Also, a great part of them (78%) are aware of the existence of software tools that help with accessibility issues. Even though, only 56%

From the open-ended questions part some points can be extracted. The respondents mention that they have two major challenges; one is to maintain a balance between accessibility and graphic design and the other is to convince clients of the importance of accessibility in a website. These two points conclude that accessibility is a matter of all the parts involved in creating a website and that web designers cannot solve this aspect alone. Many respondents claimed that their client did not want to include any accessibility aspect or did not want to spend part of their budget in this matter.

2.5 Challenges arisen

The challenges arisen from the previous study are presented in this section. These challenges are seen as improvements that can be done in one or many areas regarding web accessibility and visually impaired users.

The challenges are presented in the Table 1. The first column corresponds to the challenge and the second column corresponds to its rationale.
<table>
<thead>
<tr>
<th>Challenge</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision of WCAG 2.0</td>
<td>WCAG 2.0 seems to not completely fulfill the accessibility issues. The most low level guidelines, level-A, are not fulfilling the needs of visually impaired users. This statement can be extracted from point 2.4.2.</td>
</tr>
<tr>
<td>Encouragement of WCAG 2.0 when creating a website</td>
<td>The implementation of WCAG 2.0 is still poor in most websites. It is rare that a government or a top ranked website has more than a level-A punctuation. Level-AA and level-AAA websites were rarely found during the studies. This statement can be extracted from point 2.4.1.</td>
</tr>
<tr>
<td>Improvement of screen readers regarding the new content notification system</td>
<td>Most of the screen readers fail to notify in an efficient way that new content has appeared in the website. This does not allow visually impaired users to fully experience and benefit from dynamic websites. This statement can be extracted from point 2.4.3.</td>
</tr>
<tr>
<td>New approach on assistive technology</td>
<td>Some visually impaired users do not feel comfortable using regular screen readers. There is a need for a solution that allows customization in terms of accessibility features. This statement can be extracted from point 2.4.3.</td>
</tr>
<tr>
<td>Balance accessibility and graphical design</td>
<td>The design of moder web sites does not take into account the importance of that website being accessible. This lack of awareness may lead to a website with a good design but with accessibility barriers. This statement can be extracted from point 2.4.4.</td>
</tr>
<tr>
<td>Raise awareness of the importance of accessibility to all parts involved in the life-cycle of a website</td>
<td>Web designers and developers are not the only ones responsible for including accessibility points in the website life-cycle. There are more actors involved. The lack of awareness of how important accessibility is in a website often makes the developers skip that part. This statement can be extracted from point 2.4.4.</td>
</tr>
</tbody>
</table>

Table 1: Challenges arisen from the theoretical background study
3 Research approach

In this chapter the research approach is presented. The chapter is divided into three sections. The first section is an overview of the research project. This overview describes the motivation of the problem to be solved, the proposed solution and reasoning about this solution. The second section describes the method used to implement the solution. Finally, in the third section the validation method used to certificate the implemented solution is presented.

3.1 Research overview

In this overview section different issues are presented. The research motivation; why this research should be done. The research solution; what is being proposed. And finally the solution reasoning; why this solution is being chosen.

3.1.1 Research motivation

There are many factors that affect if a website is accessible or not. One of them is the type of impairment the user suffers from. As explained in the theoretical background, a WCAG 2.0 level-A website does not ensure that an impaired user will have a full experience on that website because some needs are not covered at all by the guidelines. The use of assistive technologies does not completely solve the problem either.

As stated on the theoretical background, the impaired users have difficulties during the process of installation of an assistive technology, this being a browser plug-in or any other kind of software. Another problem regarding assistive technologies is the price: assistive commercial software is quite expensive. This is a reason that might explain the recent popularity of open source and less expensive alternatives when choosing assistive technologies.

Another controversial point regarding assistive technologies is that visually impaired users, specifically low vision users, do not feel comfortable using screen readers. The screen readers were designed mainly for blind users and do not adapt to low vision users needs.

All these issues motivate the design and development of a solution that tries to cover them.

3.1.2 Research solution

The solution proposed has to cover all these gaps and issues in an integrated manner. The solution has to be a single unit software and not multiple software each one designed to cover a specific matter.

The software solution that can solve all these issues is a middleware. A middleware is a software that stays between two layers and forwards communication between these two layers. This communication is transparent to the both layers. In the case of this project, the middleware will act as a glue layer between an original web page and the user’s browser. Choosing a middleware as a software solution allows to integrate the different gaps in a single unit.

The main gaps or issues that the software has to fulfill are these three:

1. The middleware has to offer different types of visual adaptations for different impaired users
2. The user does not have to go through an installation process

3. The middleware has to follow the WCAG 2.0 guidelines when displaying the adapted website

The first feature aims to satisfy one of the challenges arisen in the previous section. The challenge is the need of an assistive technology able to allow the user to choose between different adaptations. With this feature the middleware can include a wide range of users with each one of them having different needs of visual adaptation.

The second feature aims to introduce the software to a wider range of users by avoiding an installation process. This process increases the difficulties for a visually impaired user since it requires an interaction of the user during the installation process, which in turn is not visually adapted. By avoiding an installation process the visually impaired user does not need a third-party and more autonomy is given to him or her.

The third feature aims to standardize the result following WCAG 2.0 guidelines. These guidelines are the accepted standard in terms of accessibility by the W3C. Studies have shown that WCAG 2.0 still does not accomplish the objective of guaranteeing a full web experience for visually impaired users. However, some of the guidelines do help in a better web experience so it is appropriate that the middleware follows those guidelines.

3.1.3 Solution reasoning

The software solution chosen to develop the middleware is a web application. This choice is due to three advantages that this kind of software provides.

Firstly, a web application does not require any custom installation process given that it relies on the web browser to run the application. By avoiding any installation process, the major feature number 2 stated in the previous sections (The user does not have to go through an installation process) it is accomplished.

Secondly, a web application can be updated and maintained in a transparent way without involving the user. The user does not have to go through any update process. This transparency grants that the latest version always is used by all users.

Thirdly, a web application is independent in several aspects. It can be used in different web browsers over the most used Operating Systems, making the target group include almost the 100% of visual impaired users that operate a computer. Also, almost all features are supported by the most popular web browsers.

After these three advantages, it can be concluded that by using a web application the installation and update process is avoided, and that it can be used in almost any computer the user may reach.

3.2 Methodology

The methodology used to develop the solution is software prototyping. The product to be developed must not be considered a final solution but a prototype. This means that it will not include all the features and all the functionality that a final product should include.

The reason of using this methodology is that the scope of this project does not include the development of a full working solution. This limitation is because of time limitations and being a bachelor project. Given those limitations, using prototyping as methodology is the best choice. This methodology allows to present an incomplete but operative model of what a final product could look like.
Another reason of using this methodology is that allows to test the middleware with the users before the final version is released. This continuous feedback may change some functionalities or requirements, and add new ones.

The prototype dimension used is horizontal. With this dimension the methodology focuses on user interaction more than low-level system functionality. This dimension is useful for getting confirmation of user requirements and system scope and to develop estimations of final system time, cost and effort.

The reason for choosing this dimension is that it allows to evaluate how the requirements are being approached. This is confirmed by direct feedback from users. With this type of feedback, the prototype requirements and features can be modified, deleted or added in a dynamic and fast way. The core of the system does not need to be changed.

The type of prototyping used is Evolutionary. The main goal of this type is to build a robust prototype in a structured manner and refine it in later iterations. The reason for creating a robust prototype and keep its structure is that what is built is the heart of the system and the improvements and further requirements will be built around it.

The goal of the practical part is to build a prototype that fulfils some basic requirements and gives a general idea of the final state of a product. The choice of this type of prototyping allows the fulfilment of the desired goal.

Following the continuous validation in form of users’ feedback that an horizontal prototype gives, a final user validation will be performed. This final validation test will deliver a wider and more detailed feedback from the users. These results will be used to determine the correctness of the final prototype.

The participants in the test will be visually impaired users. These users will be facilitated by the Terrassa School of Optics and Optometry of the university UPC (Universitat Politècnica de Catalunya).

The experiment setup is the following:

- **Step 1** The user tests the web application. The home page of the website acts as a sandbox. In this sandbox, the user can try the different adaptation features and adjust it to his or her needs.

- **Step 2** The user searches a certain information in a website through the web application. Each user gets a randomly selected website and which information to be found. There are two types of websites, the first type will be WCAG 2.0 compliant and the second type will not be compliant with these guidelines.

- **Step 3** The user answers a survey. When the user finds the information, he or she stops searching the website and answers the survey. The user is told that if he or she cannot find the specified information after several navigations through the website, he or she has to stop searching. Also, the user is told that if an error occurs and is not able to keep with the search has to stop. The survey has three parts: the first one collects information about the user, the second contains questions about the utility of the features, and the third questions about the utility of the middleware.

After all tests have been carried out, the obtained data will be analysed. This analysis will determine the correctness and validity of the solution and will also define possible directions for future work.
4 Prototyping

This chapter corresponds to the practical part of the project. First, the requirements analysis and how they were deduced is presented. Secondly, the design of the prototype with system and components overviews is presented. Finally, the final version and some examples of it are presented.

4.1 Requirements analysis

The following section presents the requirements analysis of the middleware. The section is divided into functional and non-functional requirements. These requirements are the ones that have to be fulfilled by the prototype.

The requirements originate from three different sources: the theoretical background, meetings with project coordinators, and meetings with an Optics and Optometry specialist.

4.1.1 Functional requirements

The following requirements were identified from the theoretical background and the studies made by V.L. Hanson and J.T. Richards [2] and by L. Evett and D. Brown [10]:

- Visually impaired users have difficulties identifying images and understanding them in context. Some images are displayed within text and that may cause misconceptions. Also, the alternative descriptive text (alt attribute of the img tag) is only used by the screen readers. This information could be better used.

- Some of the images are just decorative and do not provide any information to the user and sometimes mislead his or her attention.

- The study provides a deep analysis of which is the best or at least a more accessible way to display text to visually impaired users.

- In certain occasions, the visually impaired users need to filter the content of the web page. This filtering aims to show only the relevant text and part of the given web page.

From the meetings with project coordinators the following requirement was identified:

- The new features of HTML5 and modern web browsers give the opportunity for automatic and on the fly audio narration. This feature will complement the assistance of the user during the web navigation.

From the meetings with an Optic and Optometry specialist the following requirements were identified:

- Some visually impaired users feel more comfortable with a different contrast rather than the usual black over white, mainly white over black or yellow over blue.

- It is good for a user to have text that can be increased in size but sometimes this increase makes the user lose the overall context of the web page. This increase gives the user only the vision of a concrete part of the web page. To fix this, an amplifying text lens may be helpful because the user could see the text he/she wants to read in an increased size but without losing the position he or she is in the web page.
An extra emphasis must be done during the design of any solution for visually impaired users and an extra effort has to be made to make the website user friendly. For this, it would be good that the user easily can access the menu with all the features and that this can be changed in an easy and smooth way.

Table 2 shows a formal description of the identified requirements.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Summary</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Image transformation</td>
<td>System must display the images of the requested page into a more accessible way.</td>
<td>To ease the user recognize an image and understand it better in the whole context.</td>
</tr>
<tr>
<td>2</td>
<td>Image filtering</td>
<td>System must have the option of filtering the images to be shown by size.</td>
<td>To ease the user discard relevant and irrelevant images in the website context.</td>
</tr>
<tr>
<td>3</td>
<td>Text transformation</td>
<td>System must have the option of transforming the text of the requested page into a more accessible way.</td>
<td>To ease the user a better reading.</td>
</tr>
<tr>
<td>4</td>
<td>Contrast transformation</td>
<td>System must have the option of changing the contrast of the chosen website.</td>
<td>To ease the user differentiate between text and background.</td>
</tr>
<tr>
<td>5</td>
<td>Content filtering</td>
<td>System must have the option of showing in the adapted page only the main content of the requested page.</td>
<td>To ease the user identify the main content of the website.</td>
</tr>
<tr>
<td>6</td>
<td>Voice narrator</td>
<td>System must have a voice narrator to read the text of the chosen website.</td>
<td>To facilitate the user a support tool for reading and navigating through the website.</td>
</tr>
<tr>
<td>7</td>
<td>Amplifying lens</td>
<td>System must have a lens to amplify the text and images of the chosen website.</td>
<td>To facilitate the user a support tool for reading and navigating through the website.</td>
</tr>
<tr>
<td>8</td>
<td>Dynamic changes</td>
<td>System must have a menu to allow dynamic changes of the adaptation features.</td>
<td>To ease the user change between adaptations options.</td>
</tr>
</tbody>
</table>

Table 2: Functional requirements

**4.1.2 Non-functional requirements**

Table 3 contains the description of the non-functional requirements that were identified.

**4.2 Design**

The following section presents the design part of the middleware. First, there is a description of the system design in which the functional requirements previously stated are detailed. Secondly, there is a description of the system architecture in which the non-functional requirements previously stated are detailed.
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Summary</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WCAG 2.0 Compliance</td>
<td>System shall be WCAG 2.0 level-AAA compliant.</td>
<td>To fulfill with the current and standard web accessibility guidelines.</td>
</tr>
<tr>
<td>2</td>
<td>Persistent adaptations</td>
<td>System shall provide persistent adaptations through all the pages the user navigates.</td>
<td>To guarantee a good user experience.</td>
</tr>
<tr>
<td>3</td>
<td>Error tolerance</td>
<td>System shall be tolerant to its own generated errors and HTML errors.</td>
<td>To guarantee system stability.</td>
</tr>
</tbody>
</table>

Table 3: Non-functional requirements

![Diagram of System Architecture]

Figure 1: System overview

### 4.2.1 System architecture

The architecture of the middleware is designed following the software architectural pattern Model-View-Controller. This pattern was originally used for implementing user interfaces, but it is now being widely adopted as an architecture model for implementing web applications. This architecture is composed of three parts: the model, which corresponds to the data, the view, which is the representation of the data, and the controller, which manages the data. The reason of using this pattern is that it allows to divide the visual representation of the data and its management. This architecture model allows to decouple the data, its representation, and its management. This makes it possible to change one part without affecting the other two.

In the middleware, the model corresponds to all the HTML elements which are defined by tags. The view part corresponds to two php files. The first file, index.php, is the home page of the middleware. In that page the user selects the URL to be adapted and also the accessibility features to be applied. The second file, adapt.php, is the file that presents the adapted website. The controller is the part responsible of extracting the data from a given URL and make the proper changes and adaptations (see Figure 1).
4.2.1.1 Model
The model elements are manipulated by the Controller and update the View. The Model of the middleware is formed by all the HTML elements. When elements are enclosed within angle brackets they form HTML tags. The written representation of these elements is the following: `<element_name>`.

The full list and description of all HTML elements is stated by the W3C.

4.2.1.2 View

The View is updated by the Model and displayed to the user. The view of the architecture is formed by the two files index.php and adapt.php.

The first file, index.php, is the front page of the middleware. The elements of that file are:

**URL input** Text box containing the URL of the website to be adapted.

**Accessibility features menu** This menu contains the different accessibility adaptations that the user can apply. These adaptations are grouped in:

1. Images: Concerning images and its adaptation. Includes the features *image filtering* and *image transformation*.
2. Text: Concerning text and its visual effects. Includes the features *text size* and *contrast*.
3. Web content: Concerning the content to be shown from the selected web. Includes the feature *content filtering*.
4. Lens and narrator: Concerning other media effects. Includes the features *amplifying lens* and *content narrator*.

**Usage tip** Message with an image to help the user how to use the application. This tip explains to the user how to quickly change the accessibility adaptations without having to return to this front page.

The second file, adapt.php, is the file that receives the adapted website from the Controller. Once the file has received the data, the adapted web page is shown.

4.2.1.3 Controller
The Controller manipulates the model. It is formed by the file Controller.php. This file contains a content extractor class and the proper controller class.

The context extractor class uses a third-party library called simple_html_dom.php. This library is a HTML DOM parser written in PHP 5 that makes it possible to manipulate the HTML elements of a certain web page. The elements are manipulated using the same syntax as the library jQuery, this facilitates working with HTML elements. The library also supports invalid HTML code, which means that it is tolerant to HTML tag errors.

---

11 http://www.w3schools.com/tags/default.asp
12 http://simplehtmldom.sourceforge.net/
13 http://api.jquery.com/
4.2.2 Use case

The middleware only has one use case. This use case is formed by the petition of a user to adapt a certain website.

The steps that follow the use case of adapting a website are the following:

1. The user starts the web page adaptation request. This request is triggered when the user types the URL of the web page to be adapted and submits it.

2. The View submits the petition to the Controller, sending the URL and the adaptations options the user has selected.

3. The Controller sends a web data petition (HTTP GET) to the URL defined by the user.

4. The Controller receives the data as a HTML document. The HTML elements are extracted from this document and the visual adaptations are applied.

5. The View receives the adapted HTML elements and presents them as a HTML document to the user.

An UML diagram of this use case can be found on Figure 2.

4.2.3 Functional requirements design

In this subsection the design of the functional requirements previously stated is described.

Image transformation

By this requirement the user may have the option of increasing the size of the images displayed in the website or to keep its original size. Also, this requirement shall present the images in a more accessible way. This means that the images have to be displayed inside a black-bordered square and below the image a description of it is shown. This description will be the text included in the `alt` attribute of the image HTML tag.

To fulfill this, the Controller uses the method `adaptImages(filter; increase)`. The parameter `increase` determines whether the image size has to be increased or not. The
method searches all the images in the body element and puts them through a filter (explained in more detail later). The images that pass the filter are increased or not and wrapped inside a div element. This div element contains a black border and below the image the descriptive text.

Below two code snippets of the Controller are shown. The first one shows how the images are increased if the user selects it. The second code snippet shows how the image is wrapped within a black border and the description is shown below the image.

code/controller/ControllerClass.php

```php
if ($increase == "bigger")
    $imageElement->style = "width:120%;";
```

code/controller/ControllerClass.php

```php
$imageElement->outertext = "<div>" . $imageElement->outertext . "<div>
    ($imageElement->alt != '' ? $imageElement->alt : "No description")
</div></div>";
```

Image filtering

By this requirement the user may have the option of filtering the images by their original size in the website. With this, the user can choose between displaying more or less relevant images. The relevancy of the images is considered by its size in the whole context.

To fulfill this, the Controller uses the method `adaptImages(filter, increase)`. The parameter `filter` sets the filter size that will be applied during the filtering part of the method. The filter has three values `big`, `medium`, and `small`. This value is used when the method starts processing each image found on the page. For each image found its square size is calculated (height * width) and evaluated against the filter value. If the image passes the filter the processing continues, if not the image HTML tag is deleted.

Below a code snippet of the Controller is shown. The code shows how the images are filtered by its size.

code/controller/ControllerClass.php

```php
list($width, $height) = @getimagesize($imageElement->src);
if ($width * $height < $imageMin)
    $imageElement->outertext = "";
```

Text transformation

By this requirement the user may have the option of changing the size of the text. Also, this requirement shall present the text in a more accessible way. This means that the text has to be displayed in a Sans Serif font type and left-aligned. Also, the underlined and italic text has to be changed into emphasized text with the HTML tag `<em>`.

To fulfill with this, the Controller and the View are involved. The Controller applies the static changes and the View is responsible for the dynamic changes.

The Controller adds one stylesheet to the `<head>` HTML element. This stylesheet is `text-stylesheet.php` which will receive the size of the text as a GET parameter. The size is previously selected by the user in the home page of the web application or if not, set by default. The stylesheet will transform all the text into a sans-serif font, left-aligned and to
the chosen size. The HTML headings size (h1, h2, h3, h4, h5, h6) will also be increased proportionally.

The View contains JavaScript code (contextMenuLoader.js) that allows to dynamically change the text size without reloading the page. For this, the View uses the method `changeSize(value)`. This method is called when the user increases or decreases the text size from the contextual menu. The method receives a parameter with the new text size. To update the text size, first removes the previous stylesheet added by the controller (text-stylesheet.php) and replaces it for the same but passing the value of the new text size as a GET parameter. For example, being the new text size 1.4em the replacement would be `text-stylesheet.php?textSize=1.4`.

The units used for the text size are `em`'s. This unit equals to the currently specified point size of the user. Using this unit the middleware adapts to the current point size the user has and the increase or decrease is proportional to different configurations.

Below a code snippet of the Controller and the file `text-stylesheet.php` is shown. The code shows how the text size is sent as GET parameter. Following this there is code where the CSS style is added to all text tags. The fragment of the file `text-stylesheet.php` shows where the GET parameter is received and how the text size is set.

```php
//code/controller/ControllerClass.php
$this->addStylesheet("view/css/text-stylesheet.php?textSize=");
$this->addStylesheet("view/css/accessible-stylesheet.php");
.tagsToBeModified = ["p", "a", "li", "table", "span", "label"];
foreach ($tagsToBeModified as $tag)
  foreach ($this->body->find($tag) as $element)
    $element->class = $element->class . "accessible-style";

//code/view/css/text-stylesheet.php
<?php
  header("Content-type: text/css");
  include_once '../../controller/constants.php';
  isset($_GET['textSize']) ? $textSize = $_GET['textSize'] : $textSize = 1.2;
?>
/* Style for regular text */
.accessible-style {
  font: <?php echo $textSize . 'em' ?> arial,sans-serif !important;
  text-align: left !important;
}
```

Contrast transformation

By this requirement the user may have the option of changing the contrast colors of the whole adapted page. Given a certain range of contrasts the user may change it in the adaptations features selection page or dynamically when navigating.

To fulfill this, the Controller and the View are involved. The Controller will apply the static changes and the View will be responsible for the dynamic changes.

The Controller adds one stylesheet to the `<head>` HTML element. This stylesheet is `polarity.php` which will receive as a GET parameter the type of contrast selected. This
type is previously selected by the user in the home page of the web application or if not, a
default value is set (original contrast). The possible parameters values are original, simple,
inverted or blue. The value original will keep the same colors as the chosen page. The
value simple will set the background to white and all text to black. The value inverted
will set the background to black and all the text to white. And the value blue will set the
background to blue and all the text to yellow.

The View contains JavaScript code (contextMenuLoader.js) that allows to dynamically
change the contrast without reloading the page. For this, the view uses the method
changeContrast(value). This method is called when the user selects a different contrast in
the contextual menu. The method receives a parameter with the new contrast. To update
the contrast, a check if it is a change into original colors is made. If so, the stylesheet
polarity.php is removed. If it is another contrast option the stylesheet is replaced by the
same one but passing the value of the new contrast as a GET parameter. For example,
being the new contrast option blue the replacement would be polarity.php?format=blue.

The contrast combinations selection follows the WCAG 2.0 Contrast. The guideline
says that the contrast ratio must be of at least 4.5:1 [3]. The three contrast options, simple,
inverted and blue have a contrast ratio of 21:1 (simple and inverted) and 8:1 (blue).

Below a code snippet of the file polarity.php is shown. The fragment of the file con-
tains the code that receives the contrast chosen by the user as a GET parameter. After
that, it declares the color variables and sets that colors as style to the selected elements of
the HTML document.

```
code/view/css/polarity.php

<?php
    header("Content-type: text/css");
    $colorFormat = $_GET[‘format’];
    $highlightColor = "yellow";

    switch ($colorFormat) {
        case ’simple’:
            $backgroundColor = "white";
            $fontColor = "black";
            break;
        case ’inverted’:
            $backgroundColor = "black";
            $fontColor = "white";
            $highlightColor = "#FFFF00";
            break;
        case ’blue’:
            $backgroundColor = "blue";
            $fontColor = "yellow";
            $highlightColor = "#006200";
            break;
    }
    if ($colorFormat != "original") {
        body, div, a, h1, h2, h3, h4, h5, h6, span, p{
            background-color: <?php echo $backgroundColor ?> !important;
            color: <?php echo $fontColor ?> !important;
        }
    }
```
Content filtering

By this requirement the user may have the option of filtering the content of the page. This filter consists of a selection between displaying the whole page or only the main content.

To fulfill this, the Controller and the View are involved. The Controller applies the static changes and the View allows the user to change the option of this feature and automatically reload the page to display the new content.

For this purpose the Controller uses the method `adaptMainContent(type)`. This method receives a parameter. This parameter indicates whether the user chose to display the whole page or only the main content. If the method receives the option of displaying only the main content, the Controller gathers it. To consider an element of the page as main content it has to satisfy any of the following rules:

- The HTML element corresponds to a main element represented by the main tag `<main>`.
- The id of the element starts by the word `main`.

If the element satisfies at least one of the above rules then it is added to the main content list. Finally, all the items contained in that list are displayed and wrapped inside a `div` element. This `div` has a style (web-content) defined in the file `accessible-stylesheet.php`. The style defines that the content will be displayed in the center of the screen.

The View contains JavaScript code (`contextMenuLoader.js`) that allows to switch between displaying all the web content or only the main content. The page needs to be reloaded to apply this change.. This is handled by the same JavaScript code, specifically by the method `changeContent(value)`. This method is triggered when the user switches the content to be displayed in the contextual menu. The method will update the cookie storing the value of the content to be displayed and automatically reload the web page.

Below a code snippet of the Controller is shown. In the code snippet shows how the main content is filtered by the rules previously described. The content allowed by the filter is then wrapped in a `div`.

```php
if ($this->htmlContent->find("main"))
    $this->body->innertext = $this->htmlContent->find("main",0);
else
    if ($this->htmlContent->find("[id^=main]")) {
        $accumulatedMains = "";
        foreach ($this->htmlContent->find("[id^=main]") as $mainElement) {
            $accumulatedMains = $accumulatedMains . $mainElement;
        }
    }
```

Voice narrator

By this requirement the user has the option of having the text of the web page read aloud. When the mouse is over a text, it is read by the voice narrator.
To fulfill this, the Controller and the View are involved. In this case, the Controller only adds the JavaScript files needed for the View and does not take more responsibility. Given that this requirement needs more interaction with the page, the View will contain the main part of the code.

The Controller uses the method `audioGuidance(activated)`. In this method two actions are performed. First, the JavaScript code that will perform the audio narrator effect is added. This code is located in the file `textToSpeech.js`. Secondly, the Controller sets a JavaScript variable `language` which, as the name implies, will contain the main language of the web page. This language is extracted with the controller method `extractLanguage(content)`.

The View uses two files, `textToSpeech.js` and `contextMenuLoader.js`. The first file contains the voice narrator. The procedure is the following:

1. Given an array of five languages (English, Spanish, French, Italian and Deutsch), one of them is selected by checking the JavaScript variable `language`. If this variable is not available, English will be chosen by default.

2. Given an array of HTML elements (`p`, `a`, `li`, `span`, `td`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `option`, `label`, `button`), for each element an event listener is added. The event listener will be triggered when the mouse is over one of these elements. The actions performed are first, highlight the element (for a better visual identification of what is being read) and second, read the text inside that element.

The method `toggleNarrator(value)` in the file `contextMenuLoader.js` allows the user to turn the voice narrator on or off.

To implement the voice narrator the Speech Synthesis API [11] is used. This feature is currently only available in the web browsers Chrome 33 or newer, Safari 7 or newer, iOS Safari 7.1 or newer and Chrome 37 for Android.

Below a code snippet from the file `textToSpeech.js` is shown, which acts as the core part of this feature. The code snippet shows how an event listener is added to each tag containing text.
var tagsToAddSpeech = ["p", "a", "li", "span", "td", "h1", "h2", "h3", "h4", "h5", "h6", "option", "label", "button"];

// Loop for all the tag elements to add speech
for (var i = tagsToAddSpeech.length - 1; i >= 0; i--) {
    // Getting all the elements of the tag
    elements = document.getElementsByTagName(tagsToAddSpeech[i]);

    // Adding events listeners to every element
    for (var j = elements.length - 1; j >= 0; j--) {
        var defaultBackgroundColor = elements[j].style.backgroundColor;

        elements[j].addEventListener('mouseover', function (e) {
            if (window.narrator == false)
                return;
            utterance.text = this.innerText;
            window.speechSynthesis.speak(utterance);
        });

        elements[j].addEventListener('mouseout', function (e) {
            e.preventDefault();
            if (window.narrator == false)
                return;
            window.speechSynthesis.cancel();
        });
    }
};

Amplifying lens

By this requirement the user may have the option of having a lens that amplifies the text and the images when the mouse is over them.

To fulfill this, the Controller and the View are involved. Given that this requirement needs more interaction with the page, the View will contain the main part of the code.

The controller uses the method amplifyingLens(activated, speed). This method will perform the following actions:

1. Adds the stylesheet of the lens (amplifying_lens.php). This stylesheet receives the current color format as a GET parameter. This format is determined by the contrast option selected by the user.

2. Adds the files containing the JavaScript code. The files included are jquery.imageLens.js, jquery.marquee.js and amplifyingLens.js. These files will be used by the View for the dynamic actions.

3. Adds a HTML object to the body tag. This object corresponds to a div that contains the lens.

The View uses four files, jquery.imageLens.js, jquery.marquee.js, amplifyingLens.js and contextMenuLoader.js. The first two files are libraries. The file jquery.imageLens.js
corresponds to a jQuery plug-in for a lens effect image zooming.\textsuperscript{14} The file \texttt{jquery.marquee.js} corresponds to a jQuery plug-in to create a scrolling text effect.\textsuperscript{15}

The file \texttt{amplifyingLens.js} uses both of the previous libraries to create the effect of an amplifying lens to text and images. To amplify the text, an event listener is added to all the \texttt{p}, \texttt{h1}, \texttt{h2}, \texttt{h3}, \texttt{h4}, \texttt{h5}, \texttt{h6}, \texttt{a}, and \texttt{label} elements. The listener will be triggered when the mouse is over any of them. When it is triggered, a lens with the text of the element will be displayed. The lens will display the text in a horizontal scrolling effect and an incremented text size of 6 em's. To amplify the images, an event listener is added to all the \texttt{img} elements. The listener will be triggered when the mouse is over. When it is triggered, a lens will appear with a zoomed image.

Below there is a code snippet of the file \texttt{amplifyingLens.js} which acts as the core part of this feature. The code snippet shows the procedure triggered when the mouse goes over certain tags. This tags are the ones corresponding to tags including text.

```
code/view/js/amplifyingLens.js

\$('p,h1,h2,h3,h4,h5,h6,a:not(:has(img)),label').mouseenter(function (event) {
    if (window.lensActivated == false)
        return;
    \$('#lens').text($\(\textit{this}\).text());
    $\(\textit{this}\).addClass('highlight');
    yMargin = 30;
    xMargin = -50;
    \if (event.pageX < 2*\$(\textit{window}).width() / 5)
        xMargin = -100;
    \fi
    \if (event.pageX + \$(\'#lens\').width()/2 > \$(\textit{window}).width() - 2*\$(\'#lens\').width()/3)
        xMargin = -\$(\'#lens\').width();
    \fi
    \(\$\('#lens\').css({
        top: event.pageY+yMargin,
        left: event.pageX+xMargin
    }).show().marquee({duration:parseInt(\$(\textit{window}).marqueeSpeed)}));
});
```

**Dynamic changes**

By this requirement the user may have the option of changing the adaptation features without having to navigate to the home page. With this, the user can change one feature without leaving the current page and apply this change dynamically.

To fulfill this, the Controller and the View are involved. The Controller will only add the JavaScript files needed for the View. Given that this requirement needs more interaction with the page, the View will contain the main part of the code.

\textsuperscript{14}http://www.dailycoding.com/Posts/imagelens__a_jquery_plugin_for_lens_effect_image_zooming.aspx
\textsuperscript{15}https://github.com/aamirafridi/jQuery.Marquee
The Controller uses the method `addContextualMenu()`. This method adds two JavaScript files inside the `head` element. The files are `modernizr.custom.js` and `contextMenuLoader.js`. These files will be used by the view to load the options menu.

The View uses the two previously imported JavaScript files `modernizr.custom.js` and `contextMenuLoader.js`. The first one, `modernizr.custom.js`, corresponds to a feature detection library for HTML5 and CSS3. The library used by the application is custom build. It checks whether the user can use the custom context menu or not. The credit of this library corresponds to Modernizr. The second file, `contextMenuLoader.js`, uses the previously stated library to detect if the feature of the custom context menu can be used in users browser. If the feature can be used, the file `context-menu.js` containing the HTML code to create the custom contextual menu is loaded. If the feature is not supported then a polyfill is loaded. The polyfill created for the contextual menu uses the library ContextJS.

Below two code snippets of the file `contextMenuLoader.js` are shown. The first snippet details how the contextual menu JavaScript files are imported. This import differs from browsers where the contextual menu can be customized and where a polyfill must be imported. The second snippet is an example of one dynamic change that the (polyfill or browser) contextual menu allows, specifically changing the size of the text without reloading the whole web page.

```javascript
// LOAD MODERNIZR
Modernizr.load('view/languages/lang.' + language + '.js');
Modernizr.load({
    test: Modernizr.contextmenu,
    yes: 'view/js/context-menu.js',
    nope: ['libs/context.js', 'libs/context.standalone.css',
          'view/js/context-menu-polyfill.js']
});
```

```javascript
var changeSize = function (value) {
    if (value == 'less')
        textSize -= 0.2;
    else if (value == 'more')
        textSize += 0.2;
    $('link[rel=stylesheet][href^="view/css/text-stylesheet.php"]').remove();
    $('head').append('<link rel="stylesheet" type="text/css" href="view/css/text-stylesheet.php?textSize=' + textSize + '">');

    // Setting the cookie
    document.cookie = "textSize=" + window.textSize;
}
```

---

17[http://modernizr.com/](http://modernizr.com/)
4.2.4 Non-functional requirements design

In this subsection the design of the non-functional requirements previously stated is described.

WCAG 2.0 Compliance

By this requirement the middleware may pass the WCAG 2.0 level-AAA successfully. To fulfill this requirement the front page (index.php) has been designed following the WCAG 2.0 guidelines [3].

To check this compliance, a third-party tool has been used. The tool, Achecker\(^{19}\), consists on a web application that evaluates the accessibility of a given web page and displays the items or problems not compliant with the selected guidelines (in this case WCAG level-AAA).

Persistent adaptations

By this requirement the middleware may conserve the adaptations chosen by the user. This means that when the user clicks a hyperlink the following page has to be adapted with the same features as the previous one.

To fulfill this requirement the procedure is as follows. First, a session cookie is created when the user visits the front page (index.php). In this session the accessibility features chosen by the user will be stored. After that, the user types an URL and a request to adapt.php to adapt the page is made. During the process of retrieving the HTML elements of the requested website, the Controller modifies the attribute href of every anchor HTML element. The modification consists of appending the string adapt?web=URL to the URL. By this, all the links will refer to adapt.php and the web page to adapt is received as a GET parameter. Then, for every petition that the adapt.php file receives, the session features (variables) will be read from the cookie and the adaptations are made based on the options chosen by the user (see Figure 3).

Error tolerance

By this requirement the middleware may be tolerant to errors. The errors to be handled correspond to the ones generated by the middleware itself and to HTML tag errors that may come from the source web page the middleware is adapting. The errors that can occur due to dynamic code from the web page to be adapted, like JavaScript or Flash code, are not considered.

To prevent web application errors Defensive Programming techniques have been used during the coding process of the application[13].

The library PHP Simple HTML DOM used by the middleware supports invalid HTML

\(^{19}\)http://achecker.ca/checker/index.php
4.3 Implementation

In this subsection the framework used to develop the middleware and the released version is detailed. The code implementation of the solution is available online\(^\text{20}\).

4.3.1 Framework

The middleware is a web application. As a web application consists of a front-end and a back-end part. The details of the coding language used to develop each part are the following:

- **Front-end**: These are the components that the user manipulates and also the components that show the adapted contents. They are part of the View, and are developed using HTML5, CSS and JavaScript.

  The choice of using HTML version 5 is that it is the last version of this mark-up language defined by W3C. Also a release of the complete specification of HTML5 by the end of 2014 is planned by the W3C[14].

  As a complement, and for styling purposes, css-stylesheets were also included. These stylesheets were written in CSS 2.1. The reason for choosing this version and not CSS 3.0 is that the first is a Proposed Recommendation and the other is still a Candidate Recommendation, making it still not a standard[15].

  JavaScript was used to make the middleware a dynamic web application. The choice of this language is because it is the main used client-side scripting language and also it is recommended by the W3C\(^\text{21}\).

- **Back-end**: These are the components that manipulate the data and are run entirely on the server. They are part of the Controller and are developed using PHP 5.

  The choice of using PHP version 5 was based on two main reasons; the first was that the author high knowledge of the language, and the second was the usability that the library *simple-html-dom* offered when managing the HTML Document Object Model.

4.3.2 Released version

Before releasing the final version of the prototype many previous versions were designed, implemented, tested and fixed. In this subsection the details of the final release are presented. The semantic versioning used was the structure MAJOR.MINOR.PATCH\(^\text{22}\).

The version 1.0.0 is the first production ready release of the middleware. This version contains the following features: *image transformation*, *image filtering*, *text transformation*, *contrast transformation*, *content filtering*, *voice narrator*, *amplifying lens*, and *dynamic changes*.

The non-functional requirements are also included in this version. Those non-functional requirements are: WCAG 2.0 compliance, persistent adaptations, and error tolerance.

\(^{20}\)https://github.com/davidsalvador/degree-project

\(^{21}\)http://www.w3schools.com/js/

\(^{22}\)http://semver.org/
4.3.3 Examples

In this subsection some examples of the adapted web pages are shown.

Figure 4 shows the front page the user finds when accessing the middleware. It has a simple and user friendly design. There is a textbox where the user writes the URL of the web page he or she wants to adapt and two buttons. The first button will trigger the middleware to adapt the desired web page and the second button will show the features configuration menu.

Figure 5 shows the contextual menu that will appear when the user right clicks in any point of the web browser. This contextual menu displays the features configuration menu.

Figure 6 shows an original web page that can be accessed from any browser without any adaptations applied.

Figure 7 shows the same web page showed on Figure 6 but adapted by the middleware with default configuration.

Figure 8 shows the same web page showed on Figure 6 but adapted by the middleware with the contrast setting set to yellow over blue.
Figure 5: Contextual menu

Figure 6: Original web page
Figure 7: Adapted web page with default configuration

Figure 8: Adapted web page with yellow over blue contrast
5 Validation

In this chapter the validation process is detailed. The results of it are presented and analysed.

5.1 Usability test

The validation test consisted on a user usability test. This test had three parts. On the first part, the participants were free to interact with the front page of the middleware in order to discover its features and adjust them to their needs. On the second part, the participants were asked to find a certain information from a given website. This search was done using the middleware and each participant chose his or her settings. On the third part, a usability survey was made to each participant in order to receive feedback from their experience.

The participants were four visually impaired users. Those participants were facilitated by the Terrassa School of Optics and Optometry of the UPC University (Universitat Politècnica de Catalunya).

5.1.1 Conditions

The middleware version used for the usability test was the release labelled as Version 1.1. The code of this version can be found online. The test was performed during August of 2014. Each participant received an email. The email contained the test instructions and two links. The first link corresponded to the front page of the middleware, and the second link corresponded to the usability survey.

From the four users, two of them tested the same website and the other two tested another one. One website corresponded to a Spanish university, which at the time of the test was WCAG 2.0 non-compliant. And the other website corresponded to ONCE (the Spanish national blind association), which at the time of the test was WCAG 2.0 level- AA compliant. In order to check both websites WCAG 2.0 compliance, the online tool Achecker was used.

5.1.2 Usability survey

The usability survey had four questions groups, each one of them with a different aim:

- The first section aimed to gather information about the participant. An overview of the participant and its relationship with web technologies was intended. Also, a more detailed identification of its visual dysfunction was intended.

- The second section aimed to determine whether the participant was able to find the information required.

- The third section aimed to gather information about the seven middleware features. In order to achieve this, the participants were asked about the usefulness of each feature. This usefulness was to be answered using a Likert-scale between 0 and 6, being 0 the lowest value and 6 the highest.
The fourth section aimed to gather information about the middleware behaviour and its overall use. This section used a Likert-scale between 0 and 6 for some usability questions, and open-ended answers were allowed in some questions.

Table 4 shows the questions used for this survey.

<table>
<thead>
<tr>
<th>Question #</th>
<th>Question text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How old are you?</td>
</tr>
<tr>
<td>2</td>
<td>How many hours do you spend everyday on Internet?</td>
</tr>
<tr>
<td>3</td>
<td>Do you use any assistive technology? Which one?</td>
</tr>
<tr>
<td>4</td>
<td>How many years have you been suffering from low vision?</td>
</tr>
<tr>
<td>5</td>
<td>Which exact pathology do you suffer from?</td>
</tr>
<tr>
<td>6</td>
<td>Did you find the information you were supposed to?</td>
</tr>
<tr>
<td>7</td>
<td>The filtering images feature was useful</td>
</tr>
<tr>
<td>8</td>
<td>The images size transformation feature was useful</td>
</tr>
<tr>
<td>9</td>
<td>The text size transformation feature was useful</td>
</tr>
<tr>
<td>10</td>
<td>The contrast transformation feature was useful</td>
</tr>
<tr>
<td>11</td>
<td>The content filtering feature was useful</td>
</tr>
<tr>
<td>12</td>
<td>The voice narrator feature was useful</td>
</tr>
<tr>
<td>13</td>
<td>The amplifying lens feature was useful</td>
</tr>
<tr>
<td>14</td>
<td>The web application does everything I expected</td>
</tr>
<tr>
<td>15</td>
<td>The web application is simple to use</td>
</tr>
<tr>
<td>16</td>
<td>Do you think there is any feature missing?</td>
</tr>
<tr>
<td>17</td>
<td>I would recommend the web application to a friend</td>
</tr>
<tr>
<td>18</td>
<td>Any other comment?</td>
</tr>
</tbody>
</table>

Table 4: Questions used in the usability survey

5.2 Test results

In this subsection the results of the usability test are presented and analysed.

5.2.1 Participants

The information obtained from the questions 1 to 5 is presented. On that questions, the participants were asked about some information regarding their age, their relationship with websites and assistive technologies, and about their visual dysfunction.

The participants do not share a common profile but they meet at some points. There are participants who are older adults (Participant B and Participant C), there is one middle age adult (Participant A), and a young one (Participant D). Half of them (Participant A and Participant D) spend more than 3 hours a day on websites but, in contrast, those are the ones who have suffered from visual impairment for the longest time. Most of the participants (Participant B, Participant C, and Participant D) suffer from pathologies classified as severe visual impairment. Participant A suffers from a pathology classified as moderate visual impairment. However, there is one aspect in common, which is that all of them use the same assistive software; Windows Eyes.
Table 5 shows the answers to these questions.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Age</th>
<th>Hours</th>
<th>Assistive SW</th>
<th>Years of low vision</th>
<th>Pathology</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>45</td>
<td>4</td>
<td>Microsoft Lens</td>
<td>12</td>
<td>Magna myopia</td>
</tr>
<tr>
<td>B</td>
<td>65</td>
<td>2</td>
<td>Microsoft Lens</td>
<td>2</td>
<td>Glaucoma</td>
</tr>
<tr>
<td>C</td>
<td>70</td>
<td>1</td>
<td>Microsoft Lens</td>
<td>5</td>
<td>ARMD</td>
</tr>
<tr>
<td>D</td>
<td>28</td>
<td>5</td>
<td>Microsoft Lens</td>
<td>since birth</td>
<td>Retinitis pigmentosa</td>
</tr>
</tbody>
</table>

Table 5: Participants data.

### 5.2.2 Task success

The information obtained from the question 6 is presented. On that question, the participants were asked if they succeed on finding the requested information on the specified website. The possible values for this question were *Yes, I found the information*, *I found the information partially*, and *No, I didn’t find the information*.

Only participant C was unable to find the information requested. This participant had to conduct the test to a WCAG 2.0 non-compliant website. The other three participants found the whole information requested.

Table 6 shows the full answers to this question.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Information found</th>
<th>WCAG 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Yes</td>
<td>compliant</td>
</tr>
<tr>
<td>B</td>
<td>Yes</td>
<td>compliant</td>
</tr>
<tr>
<td>C</td>
<td>No</td>
<td>non-compliant</td>
</tr>
<tr>
<td>D</td>
<td>Yes</td>
<td>non-compliant</td>
</tr>
</tbody>
</table>

Table 6: Table with the information finding test data. The possible values were *Yes*, *Partially*, and *No*. The WCAG 2.0 column corresponds to whether the user was testing the compliant website or not.

### 5.2.3 Middleware features

The information obtained from the questions 7 to 13 is presented. On those questions, the participants were asked to give a score value to the middleware features. That value could go from 0 to 6, being 0 the lowest and 6 the highest score.

The *Image filtering* feature received a low value from Participant A and Participant B. However, the Participant C and D gave to the feature the maximum score value. This contrast may indicate that the feature performed well on a WCAG 2.0 non-compliant website but needs some fixing or revision when it encounters a WCAG 2.0 compliant website.

The *Image resizing* feature received a low value from Participant A and Participant B. Participant C and Participant D gave a pass value to the feature. In this case, the feature performance can be improved in order to offer a better usability to the user.

The *Text transformation* feature received the highest value from all the participants. This indicates that the feature covered a collective need and performed well in its task.
The Contrast transformation feature received a low value from Participant B and Participant D. Participant A and Participant C gave a pass value to the feature. In this case, the feature performance was in general low, given that pass and low values were given to both WCAG 2.0 compliant and non-compliant websites.

The Content filtering feature received high values from all the participants except from Participant A, who gave a low value. The mean value received is 4.75, which is a high value. This indicates that the feature performed well in mostly all the environments, and/or it was misused by one participant.

The Voice narrator feature received low values from all the participants except from Participant B, who gave it the maximum value.

The Amplifying lens feature received the highest value from all the participants. This indicates that the feature covered a collective need and performed well in its task.

Table 7 and Table 8 show the score values of the previous features.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Image filtering</th>
<th>Image resizing</th>
<th>Text transformation</th>
<th>WCAG 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>compliant</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>compliant</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>non-compliant</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>non-compliant</td>
</tr>
<tr>
<td>Mean</td>
<td>4</td>
<td>2.5</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7: Table with the features usefulness score. The lowest possible value was 0 and the highest possible value was 6. The WCAG 2.0 column corresponds to whether the user was testing the compliant website or not.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Contrast</th>
<th>Content filter</th>
<th>Voice narrator</th>
<th>Amplifying lens</th>
<th>WCAG 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>compliant</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>compliant</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>non-compliant</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>non-compliant</td>
</tr>
<tr>
<td>Mean</td>
<td>2.25</td>
<td>4.75</td>
<td>3.25</td>
<td>6</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8: Table with the features usefulness score. The lowest possible value was 0 and the highest possible value was 6. The WCAG 2.0 column corresponds to whether the user was testing the compliant website or not.

The mean score value received from the middleware features tested on WCAG 2.0 compliant websites (3.6) is lower than the score value received from the ones tested on WCAG 2.0 non-compliant websites (4.6). Being the usability score value higher in the tests run on WCAG 2.0 non-compliant websites than in the compliant ones is an unexpected result. WCAG 2.0 was strongly taken into account during the implementation process, and at a first glance, it was expected that the middleware would adapt better in WCAG 2.0 compliant environments. However, this is not a negative result but a positive one. These results indicate that the middleware adapted well to unexpected environments. Table 9 shows the mean values of the features on the different websites.
### Table 9: Table with the mean values of the features usefulness score. The values are grouped by its WCAG 2.0 compliance. The lowest possible value was 0 and the highest possible value was 6.

<table>
<thead>
<tr>
<th>Feature</th>
<th>WCAG 2.0 compliant mean score</th>
<th>WCAG 2.0 non-compliant mean score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image filtering</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Image resizing</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Text transformation</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Contrast</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>Content filter</td>
<td>3.5</td>
<td>6</td>
</tr>
<tr>
<td>Voice narrator</td>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td>Amplifying lens</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Mean</td>
<td>3.6</td>
<td>4.6</td>
</tr>
</tbody>
</table>

#### 5.2.4 Middleware overview

The information obtained from the questions 14 to 18 is presented. On those questions, the participants were asked to give a value to some general usability aspects and to give some answers to some open-ended questions. The value that the users could give was between 0 and 6, being 0 the equivalent of being totally disagree with the statement and 6 the equivalent of being totally agree with the statement.

All the participants gave the maximum accordance value to the statements *The web application does everything I expected*, *The web application is simple to use*, and *I would recommend the web application to a friend*.

Only two participants answered the open-ended questions. Participant A stated that the font size of the contextual menu was not big enough. Participant B stated that it would be easier to use the middleware if the user did not had to write the entire URL in the front page input box, and instead, that worked as a search engine.

Table 10 shows the accordance value given by the participants. Table 11 shows the answers to the open-ended questions given by the participants.
**Participant** | **Does everything expected** | **Easy to use** | **I would recommend it** | **WCAG 2.0**
--- | --- | --- | --- | ---
A | 6 | 6 | 6 | compliant
B | 6 | 6 | 6 | compliant
C | 6 | 6 | 6 | non-compliant
D | 6 | 6 | 6 | non-compliant
Mean | 6 | 6 | 6 | -

Table 10: Table with the middleware overall score. The lowest possible value was 0 and the highest possible value was 6. The WCAG 2.0 column corresponds to whether the user was testing the compliant website or not.

**Participant** | **Missing any feature?** | **Other comments** | **WCAG 2.0**
--- | --- | --- | ---
A | The font size of the contextual menu is very small | - | compliant
B | - | - | compliant
C | - | - | non-compliant
D | - | I would like to use it as a search engine instead of having to write the whole web address. | non-compliant

Table 11: Table with the answers to the open-ended questions. The questions that were not answered are represented as "-". The WCAG 2.0 column corresponds to whether the user was testing the compliant website or not.
6 Discussion

This chapter presents an overview of how the problem to be solved by this project was performed. It also compares the proposed solution with another similar research performed in the field. Finally, a review of the method used to fulfill the objectives is presented.

6.1 Problem solving

During the theoretical background study some knowledge gaps were detected, those arise some challenges for the project. This project has contributed to fill partly three of these gaps.

One challenge arisen was the need to revise WCAG 2.0 given that those guidelines did not seem to fulfill all gaps they were supposed to. The middleware offers some visual adaptations that were not covered by those guidelines. The Amplifying lens increases a text region and helps the user to keep track of the overall website. The Voice narrator reads the text where the mouse is at and assists the user through another output than visuals. Those features were identified and proposed during the meetings with the project coordinators, and with an Optic and Optometry specialist.

Another challenge arisen was the suggestion of an approach change when designing and developing assistive technologies for visually impaired users. This suggestion was in the direction of bundling different types of customizable adaptations instead of keeping the screen readers direction, and facilitate its installation. The middleware groups different customizable adaptations and acts as a web application, turning it transparent to the user and without the need of an installation process.

Finally, a balance between a web site’s graphical design and its accessibility was identified as a challenge. The middleware successfully performed, as stated in the validation process, in an accessibility-design unbalanced website. This may not be a solution to the more general problem but a small fix. So the project’s contribution to this unbalance was the middleware’s adaptation not only to WCAG 2.0 compliant websites but to non-compliant too.

Table 12 shows a summary of the project’s contribution to the identified gaps.

<table>
<thead>
<tr>
<th>Knowledge gap</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision of WCAG 2.0</td>
<td>Addition of features to cover some guideline’s gaps (amplifying lens and voice narrator)</td>
</tr>
<tr>
<td>New approach on assistive tech-</td>
<td>Use of a middleware acting as a web application</td>
</tr>
<tr>
<td>nologies</td>
<td></td>
</tr>
<tr>
<td>Balance accessibility and graphical design</td>
<td>Adaptation to WCAG 2.0 non-compliant websites</td>
</tr>
</tbody>
</table>

Table 12: Table with the knowledge gap and a summary of the contribution made by the project.

This project is not the first attempt to implement a middleware that improves web accessibility. J. Richardson and V. Hanson developed a web browser plug-in which performed visual adaptations to the websites[8].

The solution proposed by this project can be used in many browsers (Internet Explorer, Mozilla Firefox, and Google Chrome, at least). In contrast, the solution developed on the
cited study was a plug-in for a specific browser, restricting the usage of a certain browser to the user.

Another restricting aspect was the usage of a proprietary text narrator software by the cited study. The text narrator offered by the middleware of this study is based on a W3C API. This means that the W3C proposed a standard API not attached to any commercial company and can be used in different browsers.\(^{28}\)

The lack of any installation or configuration process is another difference between the middleware presented by the cited study and the one presented in this project. As previously explained, the middleware presented in this project does not need any installation process by the user. However, the solution presented by the cited study required an installation and that was remarked by the authors as a challenge to be solved.

Their solution included text and images transformations in a similar way that the presented middleware does. However, it did not include contrast transformation sets, as this project’s middleware does. With those sets (White over Black, Black over White, and Yellow over Blue) the user does not have to go through the settings many times to change the background and text color.

One great feature that their solution offered and this middleware does not is the web browser’s controllers size increment. Mouse and browser’s buttons had an increased size, making the interface of the browser more accessible. The presented middleware does not include this functionality. Given that the middleware acts as a web application it cannot handle those browser-specific settings. This feature could be included in the middleware if it was turned into a web browser plug-in, as the one presented by J. Richardson and V. Hanson.

6.2 Method reflection

The method used to implement this project was prototyping. The choice of this method allowed fast iterations between the versions and allowed to apply significant changes and features in each iteration. Another reason why prototyping method adapted well to the project was the fact that the requirements changed during the time. This changes were product of a continuous feedback with project coordinators and specialist.

Being the middleware a prototype allows to give a general idea of what the final version can be. Through the last iterations, the prototype did not experience such significant changes as it did with the first iterations. This resulted in a well-defined prototype, in terms of what visually impaired users need, but needs the improvement in the features and other aspects that a final or a stable software can offer.

\(^{28}\)http://caniuse.com/#feat=speech-synthesis
7 Conclusion

In this final chapter the concluding remarks are stated.

7.1 Conclusions

This project gave insights of the current state of web accessibility and visually impaired users. A theoretical study was made which led to a practical part which involved an implementation of a middleware and its evaluation.

Two research questions were to be answered. To answer the first question, RQ1 – How is the interaction between visually impaired users handled today?, a literature review was made. Different aspects of web accessibility and its relationship with visually impaired users were analyzed during this review; web design guidelines (WCAG), assistive technologies, and web designers. After the review, a list of challenges originated from that analysis was presented.

The second research question, RQ2 – How can the interaction between visually impaired users and websites be improved?, was tried to be answered with the design and implementation of a middleware prototype. The middleware aimed to contribute to fill three of the identified arisen challenges; Revision of WCAG 2.0, New approach on assistive technologies, and Balance accessibility and graphical design. The analysis and design of the middleware was conducted with the help of project coordinators, an Optics and Optometry specialist, and with the support of the literature review performed on previous chapters.

The answer to RQ2 raised a the subquestion RQ2.1 – Can the proposed solution improve the experience of web pages to visually impaired users?. To answer this, a usability test was made. Four visually impaired users participated on the test. The results of the test showed that some of the middleware features were really helpful to the users, like the Text transformation and the Amplifying lens, and some of them need some improvement or redesign, like Image resizing and Contrast change. As an overall result, the middleware prototype, improved the web experience to the participants but some improvement can still be done. The usability test provided valuable feedback from the participants which can be used to keep improving the middleware prototype. However, the usability results can not be extended to a wider population and generalized given that only four participants took the tests and it would require more participants to generalize the results, and also, only two websites were tested.

7.2 Further research

Given the time and load limitations of the project some further research can be done. A suggestion for further research would be a more extended and deep user usability test. This further line of research is motivated by two aims: the first is that it would give more accuracy to the usability data obtained and the second would be that it would may identify new requirements and new needs the visually impaired users have. Another suggestion for further research, related to the usability test, would be the improvement of the current features and the implementation of the needs suggested by the users during this project validation process. Finally, after this further research is completed, a stable version of the middleware could be launched and close the current prototype stage.
References


