Electronics and Computer Science

Faculty of Physical and Applied Sciences

University of Southampton

Eldar Alasgarov

27th April 2015

Automated Accessibility Tool for

Web2Access

Project supervisor: Dr David Millard

Second examiner: Dr Rong Zhang

A project progress report submitted for the award of

BSc Information Technology in Organisations

Word Count: 10684

# Abstract

The Web Accessibility Team within Electronics and Computer Science (ECS) department at the University of Southampton has a web service that assists performing accessibility checks on various websites. A software solution was developed to facilitate faster and less complicated review process of a website. A participatory spiral model was used for the project development with the Web Accessibility Team acting as a client and a participating designer. The requirements for the project were gathered from periodical meetings, which were also used for iterative design purposes. The implementation of the project was iterative as well, which allowed fast accommodation to changes and additions to the project requirements. Various front-end frameworks were used during the implementation to make the development faster and the code consistent and maintainable. The development was followed by the testing and evaluation processes where the tool received positive feedbacks as well as suggestions for future development. The output of the project fulfilled all main requirements established by the client and resulted in an offer to the researcher for further development of the prototype.

Contents

[Abstract 2](#_Toc417871855)

[List of Tables 5](#_Toc417871856)

[List of Figures 5](#_Toc417871857)

[Acknowledgements 6](#_Toc417871858)

[1. Introduction 7](#_Toc417871859)

[1.1 Problem 7](#_Toc417871860)

[1.2 Project Goals 7](#_Toc417871861)

[1.3 Project Scope 8](#_Toc417871862)

[2. Literature Review 9](#_Toc417871863)

[2.1 Accessibility over the Web 9](#_Toc417871864)

[2.2 Web2Access 10](#_Toc417871865)

[2.3 Existing Solutions 11](#_Toc417871866)

[2.4 Spiral Model Software Development 12](#_Toc417871867)

[2.5 Participatory Design Approach 13](#_Toc417871868)

[3. Project Planning 14](#_Toc417871869)

[3.1 Stakeholders 14](#_Toc417871870)

[3.2 Stakeholder Identification 14](#_Toc417871871)

[3.3 Personas and Scenarios 15](#_Toc417871872)

[3.4 Project Constraints and Potential Challenges 15](#_Toc417871873)

[3.5 Participatory Design Methodology 16](#_Toc417871874)

[3.6 Schedule 17](#_Toc417871875)

[3.7 Risk analysis 18](#_Toc417871876)

[3.8 Initial Client Requirements 19](#_Toc417871877)

[3.9 Developer Requirements 22](#_Toc417871878)

[3.10 Tools and Techniques 22](#_Toc417871879)

[3.10.1 Web Application Framework Comparison 22](#_Toc417871880)

[3.10.2 Additional Tools, Libraries, and Frameworks 23](#_Toc417871881)

[3.10.3 Development Environment 24](#_Toc417871882)

[3.10.4 Google Extension Architecture 25](#_Toc417871883)

[4. Iterative Participatory Design and Implementation 27](#_Toc417871884)

[4.1 Design Goals 27](#_Toc417871885)

[4.2 Client Engagement 27](#_Toc417871886)

[4.3 First Development Iteration 29](#_Toc417871887)

[4.3.1 Implementation 29](#_Toc417871888)

[4.3.2 Feedback Session 30](#_Toc417871889)

[4.4 Second Development Iteration 30](#_Toc417871890)

[4.4.1 Implementation 30](#_Toc417871891)

[4.4.2 Feedback Session 31](#_Toc417871892)

[4.5 Third Development Iteration 32](#_Toc417871893)

[4.5.1 Implementation 32](#_Toc417871894)

[4.5.2 Feedback Session 33](#_Toc417871895)

[4.6 Final Result 33](#_Toc417871896)

[5. Testing and Evaluation 36](#_Toc417871897)

[5.1 Testing Strategy 36](#_Toc417871898)

[5.2 Test Results 36](#_Toc417871899)

[5.3 Beta Testing 37](#_Toc417871900)

[5.4 Evaluation Goals 38](#_Toc417871901)

[5.5 Evaluation Approach 39](#_Toc417871902)

[5.6 Evaluation Results 39](#_Toc417871903)

[5.7 Future Evaluation 42](#_Toc417871904)

[6. Conclusion 43](#_Toc417871905)

[6.1 Project Goals Evaluation 43](#_Toc417871906)

[6.2 Participatory Spiral Model 43](#_Toc417871907)

[6.3 Implementation and Tools 44](#_Toc417871908)

[6.4 Schedule 44](#_Toc417871909)

[6.5 Future Work 44](#_Toc417871910)

[6.6 Conclusion 45](#_Toc417871911)

[References 46](#_Toc417871912)

[Appendices 48](#_Toc417871913)

[Appendix A: Primary Stakeholders – Personas and Scenarios 48](#_Toc417871914)

[Appendix B: Final List of Requirements 49](#_Toc417871915)

[Appendix C: Project’s Implementation History in GitHub 50](#_Toc417871916)

[Appendix D: Storyboards 51](#_Toc417871917)

[Appendix E: First Iteration Development Progress 51](#_Toc417871918)

[Appendix F: Example of Code and Algorithm – Test 2: Images 52](#_Toc417871919)

[Appendix G: Second Iteration Development Progress 53](#_Toc417871920)

[Appendix H: Third Iteration Development Progress 54](#_Toc417871921)

[Appendix I: Adjusted Criteria Text by the Client 55](#_Toc417871922)

[Appendix J: The Final Screenshots of the Google Extension 56](#_Toc417871923)

[Appendix K: Google Extension Test Cases 61](#_Toc417871924)

[Appendix L: Evaluation Tasks 62](#_Toc417871925)

[Appendix M: User Interface Problems mentioned during the Evaluations 63](#_Toc417871926)

[Appendix N: Participant Information Sheet and Consent Information 64](#_Toc417871927)

[Appendix O: Structured Interview Questions 66](#_Toc417871928)

[Appendix P: Original Project Brief 67](#_Toc417871929)

[Appendix Q: Table of Contents for the Design Archive File 68](#_Toc417871930)

# List of Tables

[**Table 1: Web Accessibility Tools** 11](#_Toc417871931)

[**Table 2: Project Stakeholders** 14](#_Toc417871932)

[**Table 3: Persona 1 - Researcher/Academic** 15](#_Toc417871933)

[**Table 4: Potential Tests for Automation** 16](#_Toc417871934)

[**Table 5: Risk Analysis** 19](#_Toc417871935)

[**Table 6: Initial Requirements** 20](#_Toc417871936)

[**Table 7: Developer Requirements** 22](#_Toc417871937)

[**Table 8: Web Application Frameworks Comparison** 23](#_Toc417871938)

[**Table 9: Tools, Libraries and Techniques Used** 24](#_Toc417871939)

[**Table 10: Formal Meetings with the Client** 27](#_Toc417871940)

[**Table 11: First Iteration Development Progress** 29](#_Toc417871941)

[**Table 12: First Additional Requirements/Adjustments to the Tool** 30](#_Toc417871942)

[**Table 13: Second Iteration Development Progress** 31](#_Toc417871943)

[**Table 14: Second Additional Requirements/Adjustments to the Tool** 31](#_Toc417871944)

[**Table 15: Third Iteration Development Progress** 32](#_Toc417871945)

[**Table 16: Third Additional Requirements/Adjustments to the Tool** 33](#_Toc417871946)

[**Table 17: Final Number of Requirements and Problems Found** 33](#_Toc417871947)

[**Table 18: Test Cases and Results** 37](#_Toc417871948)

[**Table 19: Problems Highlighted During the Beta Testing** 38](#_Toc417871949)

[**Table 20: Future Work** 45](#_Toc417871950)

# List of Figures

[**Figure 1: ‘Web2Access’ Web Service** 10](#_Toc417871951)

[**Figure 2: Spiral Model Software Development Lifecycle** 12](#_Toc417871952)

[**Figure 3: Project Schedule for the First Semester** 17](#_Toc417871953)

[**Figure 4: Project Schedule for the Second Semester** 18](#_Toc417871954)

[**Figure 5: Use Case Diagram** 21](#_Toc417871955)

[**Figure 6: Google Extension Development Environment** 25](#_Toc417871956)

[**Figure 7: Google Extension Architecture** 26](#_Toc417871957)

[**Figure 8: Sequence Diagram** 26](#_Toc417871958)

[**Figure 9: Initial Storyboards** 28](#_Toc417871959)

[**Figure 10: Final Wireframes** 28](#_Toc417871960)

[**Figure 11: Google Extension First Prototype** 28](#_Toc417871961)

[**Figure 12: Google Extension Prototype, Screenshot 1** 34](#_Toc417871962)

[**Figure 13: Google Extension Prototype, Screenshot 2** 34](#_Toc417871963)

[**Figure 14: Comparison between GE (left) and WAVE (right)** 37](#_Toc417871964)

# Acknowledgements

I would like to thank: my supervisor Dr David Millard for his constant help and support as without his help the desired project quality would not be achieved; Russell Newman and EA Draffan for their suggestions, enthusiasm and time spent participating in this project; my friends and colleagues from the course for the support and ideas; the evaluators and testers for their time and advices given for future development; and finally, my family for their constant belief in me.

# 1. Introduction

## 1.1 Problem

Web2Access[[1]](#footnote-1) is a web service developed by the Web Accessibility Team in the Electronics and Computer Science (ECS) department at the University of Southampton. Web2Access allows users to test a web service (Web 2.0) based on the variety of tests set forth by web accessibility guidelines such as WCAG 2.0 (Web Content Accessibility Guidelines) and submit their evaluation results to the Web2Access server for public access. This is explained in more detail in **Section 2.2: Web2Access.**

Currently, each test contains an exhaustive list of methods on how users can perform and assess each check based on personal opinions. This requires a significant amount of **effort** on the part of the user: download and install a variety of tools, use different browsers and screens, etc.

Because of the variety of methods provided and their inconsistency, an *experienced* user often spends a significant amount of **time** to assess a website, whereas an *inexperienced* user just feels lost among all these methods due to a **lack of knowledge**.

The problem grows when the number of websites to be checked is greater than one and the user becomes **demotivated** by routine tasks. Currently, there is no software solution to support all these checks and merge them.

## 1.2 Project Goals

Based on the problems discussed, there are five main project goals to be achieved:

* **Minimize the effort required to perform the tests:** Provide a software solution in the form of a Google Extension (GE), which will follow a user throughout the website browsing. It will allow the user to start the evaluation, perform checks, and submit the results to the Web2Access server.
* **For the experienced users, speed up the process; for the inexperienced users, make it actually feasible:** The tool must be easy to install and use by a variety of users, especially non-technical ones. It should significantly reduce the amount of time spent on checking a website and be easy as well as intuitive.
* **Tool specificity and completeness:** The tool should not be worse in its functionality than any other tool in terms of finding and highlighting web accessibility problems associated with a website. The tests will be performed either semi-automatically by the tool or manually by the user.
* **More participators with Web2Access:** The client is eager to see more contributors to the online web accessibility testing and hopes that the new solution will help increase the number of evaluators. This is a high-level business goal.
* **Balance between automatic and manual work:** A mark suggesting system should be available to the user to make it easier to decide on the appropriate score. This research goal emerged during the background research on Web2Access and will be explained in **Section 2.2: Web2Access.**

## 1.3 Project Scope

The tool will run in a user’s web browser as a GE and will communicate with an existing server located in the ECS. The tool will be designed for use by academics in schools and universities, web developers/designers, and ordinary users (without disabilities themselves) interested in web accessibility, such as parents of students with disabilities. Because of the project’s time constraints, not all checks/tests will be implemented in the GE. Only five out of 15 tests will be fully developed as well as the complete back end of the system. If, in the future, additional checks will be considered for addition to the GE, the developers should not be worried about the system’s architecture; i.e., client-server interactions.

# 2. Literature Review

## 2.1 Accessibility over the Web

The World Wide Web has become one of the most important methods of information access and communication. In the past ten years, the number of online users has increased from one billion to three billion (Fisher, 2010). However, the more people connect to the Internet, the more accessibility problems arise among those who are partially disabled. According to the World Report on Disability, it is estimated that between 15% and 19% of the earth’s population has some form of disability (Pascual et al., 2013). Despite these significant figures, the percentage of web developers who perform accessibility checks on their web services is only 5% (Trewin et al., 2010). Moreover, according to recent studies, only 6% of all organisations have a specific budget allocated for accessibility development (Bailey & Burd, 2006). There are several reasons behind this low awareness among developers and companies, such as:

* Lack of guidelines in the accessibility areas, technical constraints which are imposed on a site (e.g. dynamic graphs), decrease in creativity, or simply a general lack of accessibility awareness (Brophy & Vraven, 2007).
* Lack of development time, accessibility requirements conflicting with user requirements (Trewin et al., 2010).

Before discussing *web* *accessibility,* it is important to describe what the term actually means. The Web accessibilityrefers to the access of information for everyone with a focus on partially disabled people (Maeda et al., 2004). Web accessibility, however, is often confused with *web usability.* The former aims to remove any technical barrier that prevents users with disabilities from accessing information, and the latter aims to make overall user interaction with the system more effective and efficient (Leporini et al., 2006). In other words, accessibility is a single aspect of usability on the web. Unfortunately, developers often forget about the “side benefits” of accessibility and how it improves the overall usability of a site for every user (Pascual et al., 2013). An example of these kinds of benefits would be subtitles for videos. Such a feature is beneficial not only for people with hearing problems to access the content, but also for ordinary people who are in a loud environment and cannot hear the audio. **‘***For people without disabilities, technology makes things convenient. For people with disabilities, it makes things possible***’** (Henry et al., 2014)**.**

To standardize accessibility on the web, a variety of guidelines have been established that describe how to develop accessible websites and provide content for them. Examples are the World Wide Web Consortium (W3C) WCAG 1.0 and 2.0 developed by the Web Accessibility Initiative, Section 508 of the Rehabilitation Act Amendment, and IBM Corporate Instruction 162 (Brewer, 2004). Despite having similar contents, they have different purposes. For example, WCAG provides a set of “recommendations” that a web service should provide to be accessible, whereas Section 508 is a law under the jurisdiction of United States government and is compulsory to be applied.

On the other hand, full compliance with accessibility guidelines does not necessarily mean that a service will be fully accessible. For example, WCAG recommends providing *alternative* text (i.e., ALT tag) for every image used. Even though it is possible to automatically check for the existence of ALT text for an image, it is not possible to automatically check whether the alternative text provided is appropriate or not (Maeda et al., 2004); e.g., ‘*photo*’ or ‘*006578.jpg*’. Moreover, these guidelines provide an exhaustive list of recommendations and rules to be followed, which is not always easy to access for a nontechnical person.

## 2.2 Web2Access

The Web2Access project is an attempt to solve the problem of the guidelines being too technical. It has summarized recommendations and rules into 15 key tests that allow a user to evaluate a website against common web accessibility problems (Wald et al., 2012),(see **Figure 1**). Each test contains a variety of methods and advice on how a particular test can be accomplished. These methods usually are automated tools. Web2Access allows users to test any Web 2.0 site against a series of checks which are linked back to guidelines such as WCAG or IBM. More than 200 well-known web services, such as Amazon, Dropbox, and Facebook, have already been evaluated, and results are stored on the Web2Access website for public access (Wald et al., 2011). These data are meant to make it easier for people to choose the most accessible websites for a particular software type (e.g. audio/video tools, data storage) or for a group of people with specific disabilities (e.g. blindness and visual impairments, dexterity).



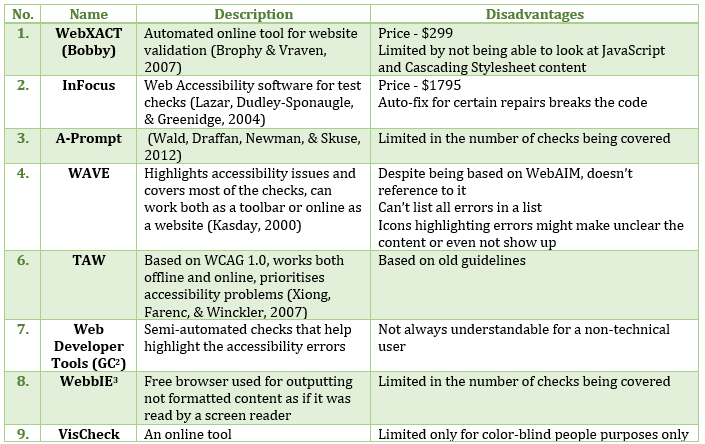
**Figure 1: ‘Web2Access’ Web Service**

Web2Access helps evaluators test a website and score it based on the results provided from the accessibility tools. However, Web2Access’s attempt to solve problems with excessively technical guidelines misses its goal. Evaluations are based on a 4-scaled marking system, where 0 is *Poor* and 3 is *Excellent.* However, it was proved that individual judgments of severity are highly personal and subjective (Brajnik et al., 2010), for example, when a person cannot decide between 2 or 3. Additionally, it has been suggested that to reduce the gap between expert and nonexpert evaluators, the severity rating must be a priority (Brajnik et al., 2011).

A potential solution for this problem could be having the GE point to a calculated and recommended score without enforcing it. A nonexpert/novice user may need to fall back on the suggested score, whereas expert users may need to give reasons why they disagree with the suggested score. This would also potentially solve additional problems related to web evaluation such as producing ‘*false* *positives*’ (i.e., finding wrong problems) or missing the actual problems (Brajnik et al., 2010). This will be the **fifth** and **final** research goal of the project.

## 2.3 Existing Solutions

It is often argued that the main challenge for web designers/developers ‘will not be the actual standards so much as finding tools to measure compliance’ (Craven, 2006). The number and variety of accessibility tools are impressive, and they are often divided into categories such as automatic/semiautomatic or downloadable/online/in-browser. **Table 1** lists the most popular web accessibility tools used today.



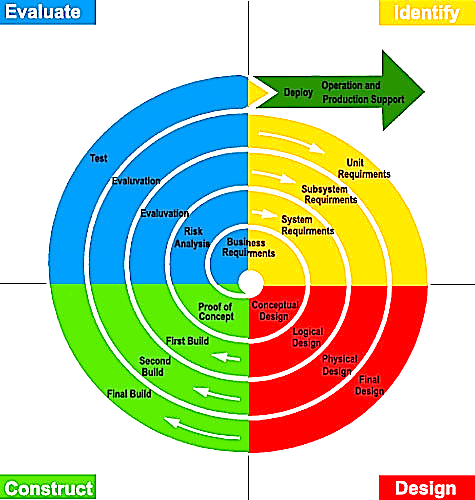
**Table 1: Web Accessibility Tools**

In addition to the disadvantages mentioned in **Table 1**, there are those that are common among all accessibility tools. Assistive technology lags in its design; by the time accommodations are made, technology has already moved another step forward (Dobransky & Hargittai, 2006). Moreover, most developers currently rely too much on the tests and forget about the human factor. The goal of an accessibility tool is not to totally replace a human evaluator, but to help them highlight existing problems on a web service in a consistent way and allow the evaluator to make a decision on the results (Leporini et al., 2006). Effective evaluation, therefore, requires a combination of both an automated tool and a person with knowledge of the accessibility guidelines (Brewer, 2004). It has been estimated that more than half the criteria and guidelines mentioned require a human assertion to be evaluated (Trewin et al., 2010).

One of the goals of this project is to create a tool that follows along with a user on the web, highlights accessibility problems, and allows the user to make the final decision.

## 2.4 Spiral Model Software Development

The spiral model, a risk-driven and iterative approach to the software process (Boehm & Bose, 1994), was chosen as the most suitable software development lifecycle process for this project. There are two main advantages of this model. One is the *iterative* approach for a growing system where the feedback from one phase provides information and decisions for the next phase (see **Figure 2**). The second advantage is a periodic and constant set of milestones for meetings to ensure stakeholder satisfaction and collect their feedback and suggestions (Mujumdar et al., 2012).



**Figure 2: Spiral Model Software Development Lifecycle[[2]](#footnote-2)**

The other advantages of the spiral model are that it includes the best parts from the *Waterfall* and *Prototyping models*, reduces the cost of later changes to the requirements in the project, and highlights the risks early in the project. The main disadvantage, however, would be the constant iteration with the client, which could be time consuming and hard to perform with only one developer.

A better solution for this project could arguably be the Agile software development process. This process is focused on incremental delivery and reduced documentation, which reduces the cost of changes throughout the project (Highsmith & Cockburn, 2001). However, the main advantage and strength of the Agile method lies in the high-quality interaction between the members of a team of developers (e.g., Scrum) which is not applicable to the current ‘*one-man team*’ project (Nerur et al., 2005).

## 2.5 Participatory Design Approach

The advantage of the current project is that it allows engagement with real clients; therefore, a *participatory* spiral model (Hansen, 2004) was proposed for the project design and development. The spiral model, as was mentioned previously, highlights the necessity of user participation and surpasses the methodologies that leave users behind in the requirements analysis/gathering stage. Additionally, the participatory design approach will increase client satisfaction by bringing them onto the design team as equal members (Millard et al., 2009). Moreover, the approach will be extended throughout the project because the client will be involved not only in the design, but also in the requirements gathering/analysis, implementation, testing, and evaluation. This will heavily depend on good coordination and relationships with the client.

# 3. Project Planning

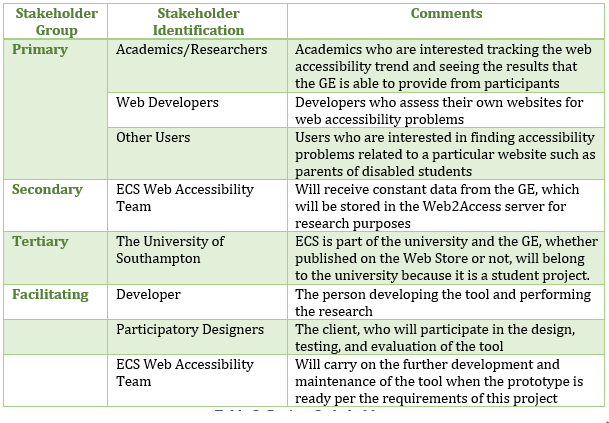
## 3.1 Stakeholders

The word *stakeholder* refers to persons, groups, or organizations whose input must be considered by the developers and that are affected by the success or failure of the system (Bryson, 2007). Moreover, stakeholders can be classified according to the level of interaction they have with the system. The stakeholders are divided into four main groups (Millard et al., 2009):

* *Primary Stakeholders*—people who actually use the system.
* *Secondary Stakeholders*—people who do not directly use the system, but who receive output from it and/or provide input to it.
* *Tertiary Stakeholders*—people who do not fall into either of the first two categories, but are directly affected by the success or failure of the system.
* *Facilitating Stakeholders*—people who are involved in the design, development, and maintenance of the system.

## 3.2 Stakeholder Identification

The tool is intended for a variety of users, regardless of their technical background. The primary users are divided into three main categories: researchers/academics, developers, and general users who are interested in finding accessibility problems related to a particular website such as parents of disabled students. These and other stakeholders are summarized in **Table 2**.

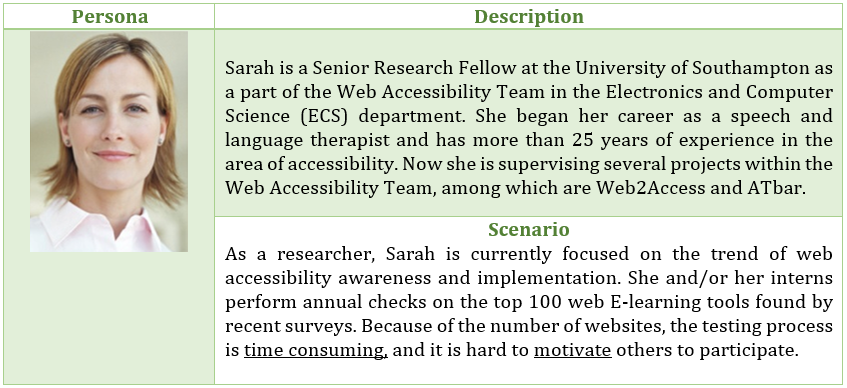


**Table 2: Project Stakeholders**

## 3.3 Personas and Scenarios

During the gathering and analysis of the requirements, the *primary* stakeholders were the main targeted user group. Analysing this category of users allowed to highlight the main problems they encounter when evaluating web accessibility. Later in the project, this helped us identify the first set of requirements for the system. The first type of primary stakeholders, *academics*, is described in **Table 3.** The other user groups can be found in **Appendix A.** The main problems faced by all three groups are summarised below.

* *Researchers* and *academics* struggle to perform checks on a large number of websites and motivate participants to do so.
* *Developers* are required to download and install a variety of sometimes unnecessary software and tools.
* *General users* lack knowledge of technology/guidelines and therefore sometimes cannot distinguish between accessible and inaccessible webpage elements.

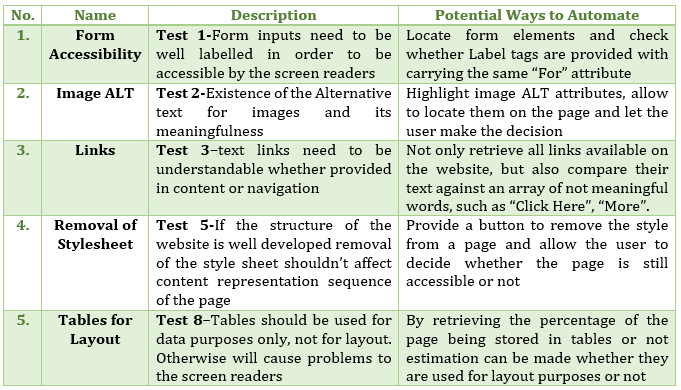


**Table 3: Persona 1 - Researcher/Academic**

## 3.4 Project Constraints and Potential Challenges

As mentioned in **Section 1.3:** **Project Scope**, not every feature will be feasible for implementation at the available time. For this reason, functional requirements that are considered *optional* will be implemented only if additional time becomes available. However, the main constraint will be the number of accessibility tests that must be implemented. Only five out of 15 available tests from Web2Access will be fully developed. **Table 4** contains information regarding each test—their name, description, and potential methods for automation of the test.

***Note:*** The data for the **Table 4** were retrieved from the Web2Access website, [www.web2access.org.uk/test](http://www.web2access.org.uk/test).



**Table 4: Potential Tests for Automation**

Moreover, the biggest potential challenge within the project could be *user motivation.* After several website tests, a user could become demotivated. A potential solution to this problem could be the addition of gamification elements to the tool by providing users with small awards (e.g., badges) and allowing them to view their ranking in comparison with others in the same group or category. Other project-associated risks and challenges are described in **Section 3.7: Risk Analysis.** Moreover, if users still abandon the tests, it would not be efficient if the important data that had been collected thus far were lost; a way could be provided to store the data in the browser to be finished later.

## 3.5 Participatory Design Methodology

One of the distinguishing features of the participatory spiral model is its emphasis on constant user participation in the project, which was strictly followed throughout this project in order to ensure and maximize client satisfaction. The developer worked closely with at least one member of the ECS Web Accessibility team (i.e., the client), and the other members of the team were asked to join when needed based on their roles, skills, and expertise. In addition to constant communication via social media, seven formal meetings were held between the developer and the client:

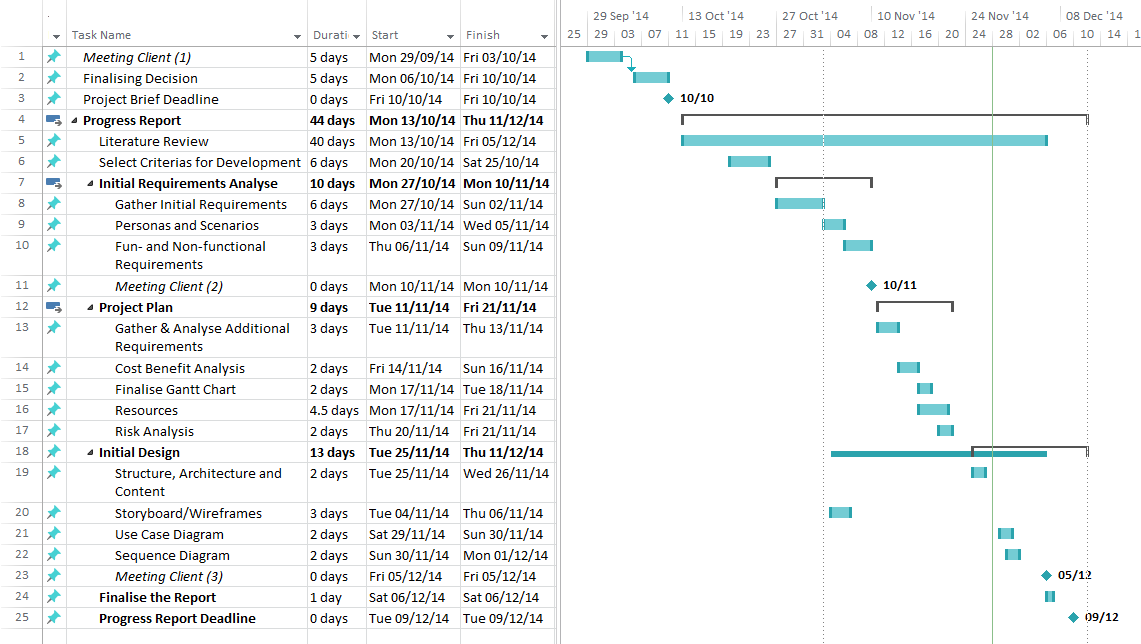
* Three for initial requirements gathering and design discussions
* Three during iterative development of the tool
* One for beta testing in the field

The *when, how,* and *what* wasdiscussed during these meetings are formalised and described in **Section 4.2: Client Engagement.**

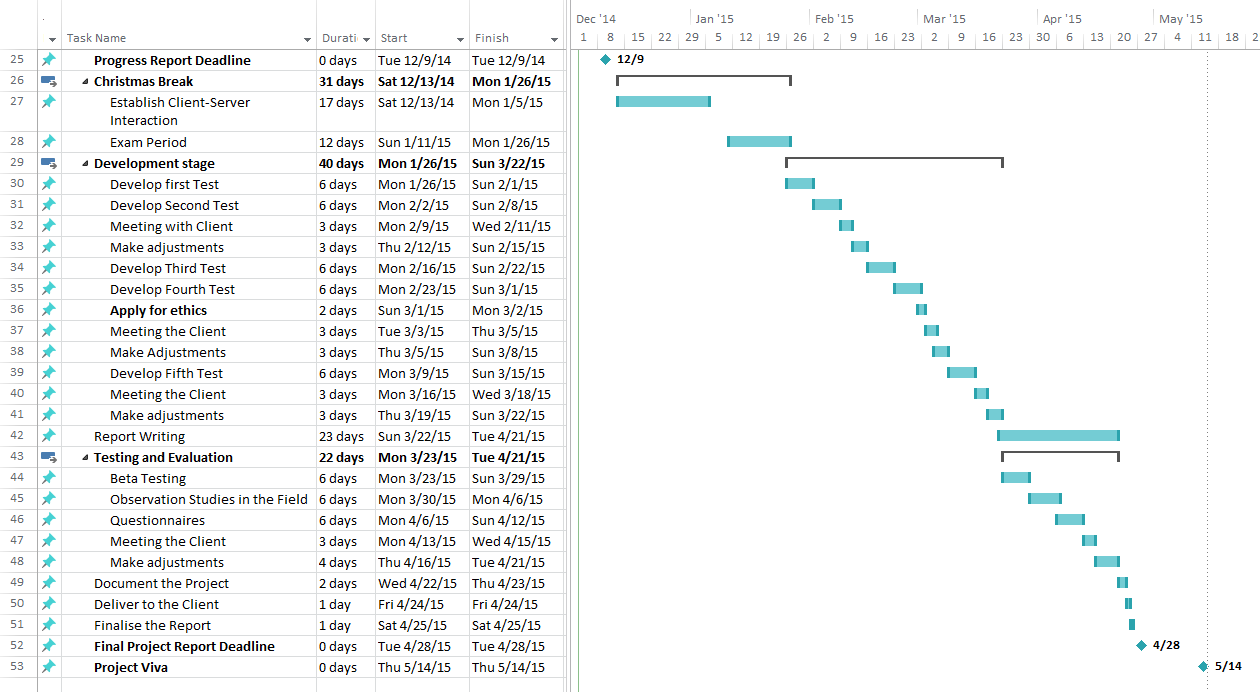
## 3.6 Schedule

The Gantt chart in **Figure 3** and **Figure 4** represents the project schedule over a year. The project was separated into two main stages. The first stage consisted of requirements gathering, design, and project planning. The second term was mostly filled with implementation, testing, and evaluation.

The project schedule was divided into weeks and days; the project duration was approximately 32 weeks or 228 days. Although strict deadlines and milestones were established, the project schedule allowed flexibility in dates in order to be more realistic.

Meetings were held periodically throughout the project to discuss and highlight the progress being made.

**Figure 3: Project Schedule for the First Semester**



**Figure 4: Project Schedule for the Second Semester**

## 3.7 Risk analysis

One of the advantages of the spiral model was the introduction of a new component, the risk assessment (Mujumdar et al., 2012). Having a *risk analysis* stage built into the model avoids many other difficulties encountered by models such as *Waterfall* (Say-Wei & Muruganantham, 2000). Prior to the commencement of the project, various risks were identified, analysed, and prioritised, and management and prevention strategies were developed. Risks are summarized in **Table 5**.

* Probability (**P**): **very low** (1), **low** (2), **moderate** (3), **high** (4), or **very high** (5)
* Severity of risk (**S**): **catastrophic** (5), **serious** (4), **tolerable** (2), or **insignificant** (1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Risk | P | S | Score P x S  (Max 25) | Action to Prevent/Manage the Risk |
| 1. | **Lost data**, **documentation,** and **duplication** in backups | 2 | 3 | 6 | **Prevent**: Provide periodic backups of the documents, once a week in two places.  **Manage**: Start from the latest backup. |
| 2. | **Project loss** (code)/server failure | 2 | 3 | 6 | **Prevent:** Use GitHub to store the history of changes.  **Manage:** Download the latest version from GitHub and start over. |
| 3. | **Requirements being added or updated** | 1 | 3 | 3 | **Prevent:** Provide constant meetings with the client (e.g., weekly).  **Manage:** Allow flexibility in time and code to actually implement the requirement(s). |
| 4. | **Size underestimation** - scope of the project | 2 | 4 | 8 | **Prevent:** Write in detail the feasibility of the project to be developed, implement part of the checks.  **Manage:** Be realistic and provide time flexibility for each stage. |
| 5. | **Time underestimation** - failure to meet deadlines | 2 | 4 | 8 | **Prevent:** Be realistic and provide time flexibility for each stage.  **Manage:** Reduce the number of requirements according to their priority. |
| 6. | The client will not **approve** the final design/product | 2 | 3 | 6 | **Prevent:** Follow participatory design actions.  **Manage:** Agree on a final design, so if the client doesn’t agree with final result, the initially agreed requirements list could be presented. |
| 7. | **Feasibility** to automate the **test** may fail | 3 | 4 | 12 | **Prevent:** Research every test and ensure that there are ways to perform this test.  **Manage:** Have a backup test suitable for replacement in case of failure. |
| 8. | **Feasibility/ limitations** of the GE | 3 | 4 | 12 | **Prevent:** Conductin-depth research on the variety of tools and techniques to be used.  **Manage:** Maintain possible alternatives to the technologies used**.** |
| 9. | **Security**, the results submitted to the server | 3 | 5 | 15 | **Prevent:** Consider every possible vulnerability and prevent it; double check with a university representative.  **Manage:** Maintain a security plan in case of an attack. |

**Table 5: Risk Analysis**

## 3.8 Initial Client Requirements

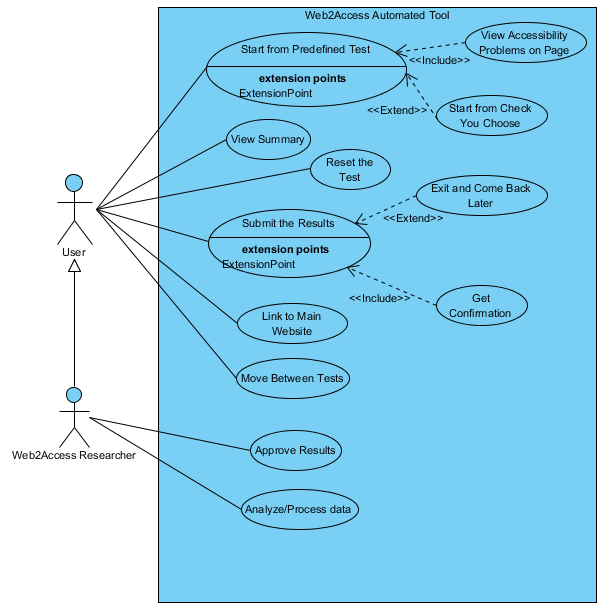
In order to begin the first development iteration in semester two, a list of initial requirements was presented. For this to occur, *three* initial meetings were held with the client, each of which produced a set of requirements, which were added to the list, as well as feedback. The process of meetings with the client is explained in detail in **Section 4.2: Client Engagement**. Functional and non-functional requirements were derived as a result of these three meetings and are listed in **Table 6.** This short list of the requirements will grow during the development iterations mentioned in **Section 4: Iterative Participatory Design and Implementation.** A full list of the project requirements can be found in **Appendix B.**

|  |  |  |
| --- | --- | --- |
| ID | Name | Date |
| M1 | First Meeting | **29/09/2014** |
| M2 | Second Meeting | **10/11/2014** |
| M3 | Third Meeting | **05/12/2014** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Requirements and Description | Priority | Done |
| FR1 | **5 Tests** | Provide at least five various checks/tests demonstrating full functionality and communication with the server | **Compulsory** | **Yes** |
| FR2 | **Move Between Tests** | Ability to easily move between checks (e.g., Back/Forward arrows) | **Compulsory** | **Yes** |
| FR3 | **Test Results** | A way to view statistics and examples of performed test results | **Compulsory** | **Yes** |
| FR4 | **Highlight DOM** | For DOM (Document Object Model) related checks, allow users to view/highlight the specific element on the page by hovering over it from the extension | **Compulsory** | **Yes** |
| FR5 | **Out of Sight DOM** | Show with arrows on the page if the DOM element is outside the browser window, requiring the user to scroll down | **Optional** | **Yes** |
| FR6 | **Marks Summary** | View the summary of progress for the checks on a specific website | **Compulsory** | **Yes** |
| FR7 | **Reset Button** | To delete the test or all tests | **Compulsory** | **Yes** |
| FR8 | **Link to Web2Access** | Provide links to the main website (Web2Access), either for reference purposes or for more detailed information | **Compulsory** | **Yes** |
| FR9 | **STEAM-Reader** | Add STEMReader for checking the MathML content of the website | **Optional** | **No** |
| FR10 | **Contact Details** | Collect optional user contact information | **Optional** | **No** |
| FR11 | **Tabbing** | Allow tabbing for non-mouse access | **Compulsory** | **Partly** |
| NF1 | **Short Criteria** | Reduce information presented to the user per test/evaluation as much as possible due to the small sizes of GEs | **Compulsory** | **Yes** |
| NF2 | **Gamification** | Badges for those who have completed several tests and comparison to others | **Optional** | **No** |
| NF3 | **Colour Scheme** | Support the colour scheme presented on the Web2Access website | **Compulsory** | **Yes** |
| NF4 | **Tool Accessibility** | Being an accessibility tool, it has to be accessible itself | **Optional** | **Partly** |
| NF5 | **Speed** | Response time/performance—facilitate faster testing | **Compulsory** | **Yes** |
| NF6 | **Adaptability** | Easy integration of additional tests in future development | **Compulsory** | **Yes** |
| NF7 | **Sequence of Tests** | Easy to automate tests first followed by the rest | **Optional** | **Yes** |

**Table 6: Initial Requirements**

Based on the functional and non-functional requirement gathered during the requirements analysis as well as the personas and scenarios analysed, a use case diagram was created to understand the scope of the project and its functionality (see **Figure 5**).

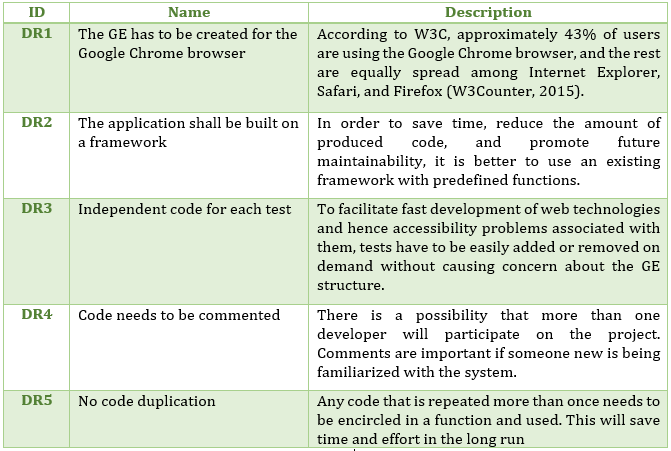


**Figure 5: Use Case Diagram**

As mentioned in the diagram, the Web2Access research group should be able to not only use the tool as an ordinary user but also access the results behind them, approve them, and process the data.

## 3.9 Developer Requirements

As a facilitating stakeholder in this project, the developer also has a set of requirements that must be implemented. This is required due to the fact that the project has potential for future development and there is a possibility that more than one developer will contribute to the development of this product. The summary of these requirements is presented in **Table 7**.



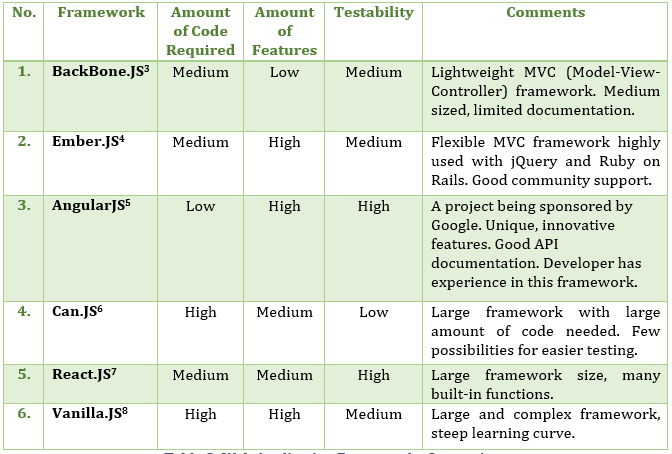
**Table 7: Developer Requirements**

## 3.10 Tools and Techniques

### 3.10.1 Web Application Framework Comparison

Because of developing a GE, JavaScript was chosen as the main programming language for this project. Because JavaScript is an object-oriented programming language, unlike Java or C#, it does not encourage structured programming. Instead, it focuses more on providing flexibility (Richards et al., 2010). This is why it is widely used as a client-side programming language for web development. Moreover, a framework for JavaScript was used in order to save time, reduce the amount of coding, and allow easier code maintenance, which will be easier to debug (Palmieri et al., 2012).

**Table 8** contains a summary of the comparison of frameworks held prior to the project implementation stage.

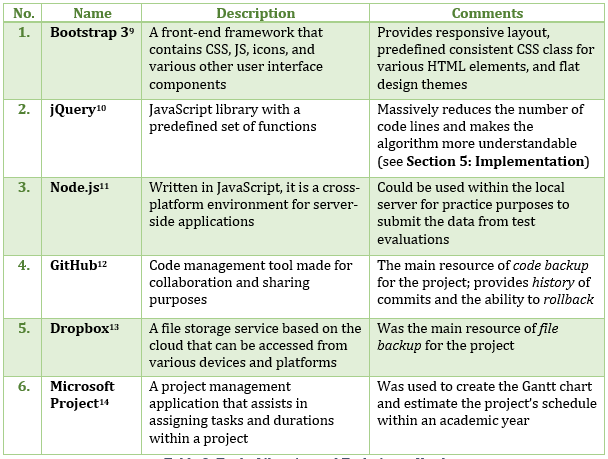


**Table 8: Web Application Frameworks Comparison**

AngularJS was chosen as the main development framework because it had the least amount of code to be written, good API documentation, and a number of unique features. Additionally, from the forums and websites visited, it was concluded that AngularJS has a bigger community of StackOverflow questions, YouTube videos, GitHub contributors, and (most importantly) GE users than any other JavaScript framework (Shaked, 2014).

### 3.10.2 Additional Tools, Libraries, and Frameworks

In addition to the JavaScript framework mentioned, other tools, libraries, and techniques have been used during the project development. They are summarized in **Table 9.**

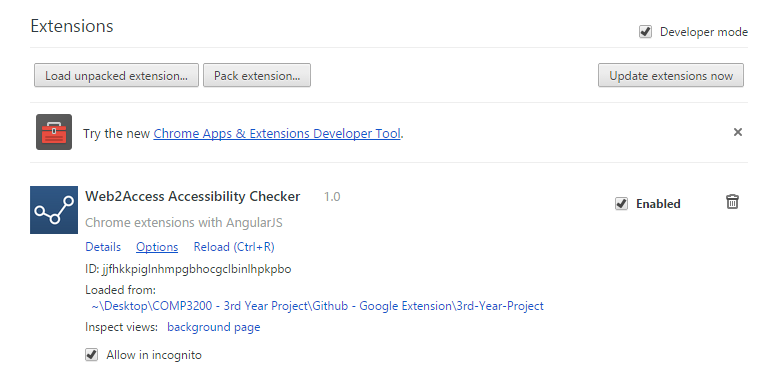


**Table 9: Tools, Libraries and Techniques Used**

### 3.10.3 Development Environment

Google Chrome Documentation states, ‘*Extensions are small software programs that can modify and enhance the functionality of the Chrome browser’[[3]](#footnote-3).*  The GE was built using HTML, CSS, and JavaScript on a developer’s local laptop by using the ‘*developer mode*’ option within the installed Google Chrome browser (see **Figure 6**)**.** From there, it could later be easily published to the *Chrome Web Store* for public use when the GE was tested and approved. Google’s online API[[4]](#footnote-4) documentation has an extensive and step-by-step list of tutorials and functions with related examples that were used during the development process. Sublime Text Editor[[5]](#footnote-5) was used as a coding tool because of its lightweight eye-friendly interface for web-related applications.

Moreover, the development folder was synced with private online GitHub repository[[6]](#footnote-6) for tracking, backup and rollback purposes. Example sections from the project’s implementation history in GitHub can be found in **Appendix C.**

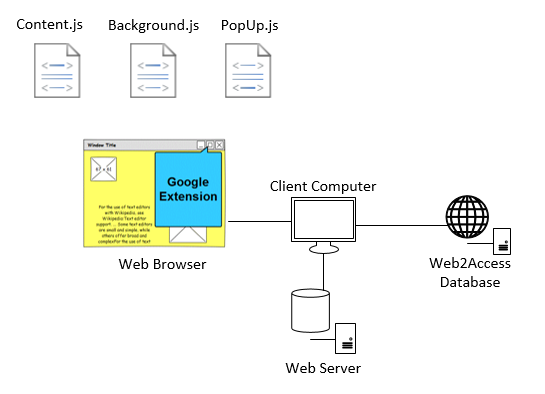


**Figure 6: Google Extension Development Environment**

### 3.10.4 Google Extension Architecture

The architecture of GEs is not complex. Apart from HTML and CSS files within the system, a developer interacts with three different .js files in three different levels (Karim, Dhawan, & Ganapathy, 2014), (see **Figure 7**):

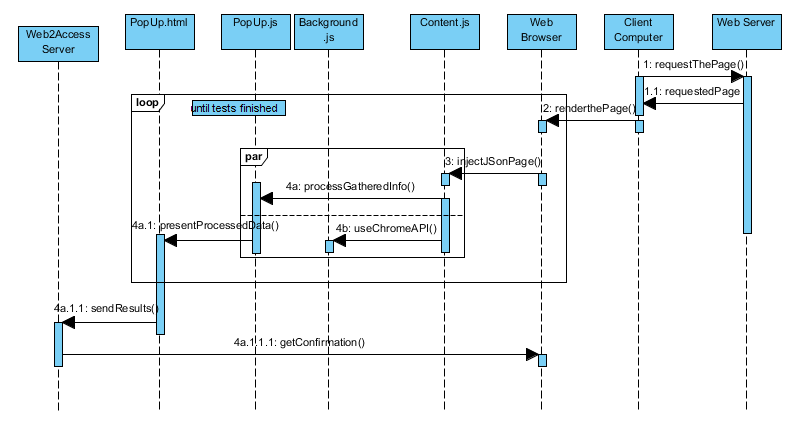
* Popup.js—This file handles the processing within the GE itself and sends messages or communicates with the active webpage via actions (e.g., press a button) and presents the data collected from a page to a user.
* Content.js—This file is embedded in the active webpage when the GE first runs. It interacts with the webpage’s DOM (Dynamic Object Model) but has limited access to the Chrome browser’s APIs.
* Background.js—Unlike the other two files, this file cannot interact with the active webpage, but has full access to the Chrome browser’s APIs. This file was introduced during implementation of **FR13,** where the developer needed to access Chrome’s Text-to-Speech function, chrome.tts. More details on this are found in **Section 4.5.1: Implementation.**



**Figure 7: Google Extension Architecture**

The rest of the architecture is quite straightforward. The *browser* on the *client’s computer* makes a request for a webpage from a *web server.* When the page is loaded, the user performs tests and marks the page based on the results he/she received. These marks will be sent to the *Web2Access* server after being submitted by the user.

To explain the architecture and sequence of actions in a more detailed manner, a sequence diagram was created in **Figure 8**.



**Figure 8: Sequence Diagram**

The process of *code injection*, *processing* *gathered data,* and *presenting* to the user will continue, or loop, until the user performs all five tests. Only after finishing the tests will all the data be sent to the server in one request.

# 4. Iterative Participatory Design and Implementation

## 4.1 Design Goals

The main focus during the design and implementation were the research goals established at the start of the project. Apart from them, there is a set of usability goals (Yvonne et al., 2007) that was considered with the same priority as the research goals.

* **Efficiency—**support the user in carrying out the tasks
* **Learnability** (easy to learn)—simple design
* **Utility**—at the appropriate level
* **Maintainability**—easily maintain and extend the functionality

Both research and usability goals (except maintainability) will be tested during the evaluation period of the project. Maintainability cannot be tested because the evaluators will not have access to the main code.

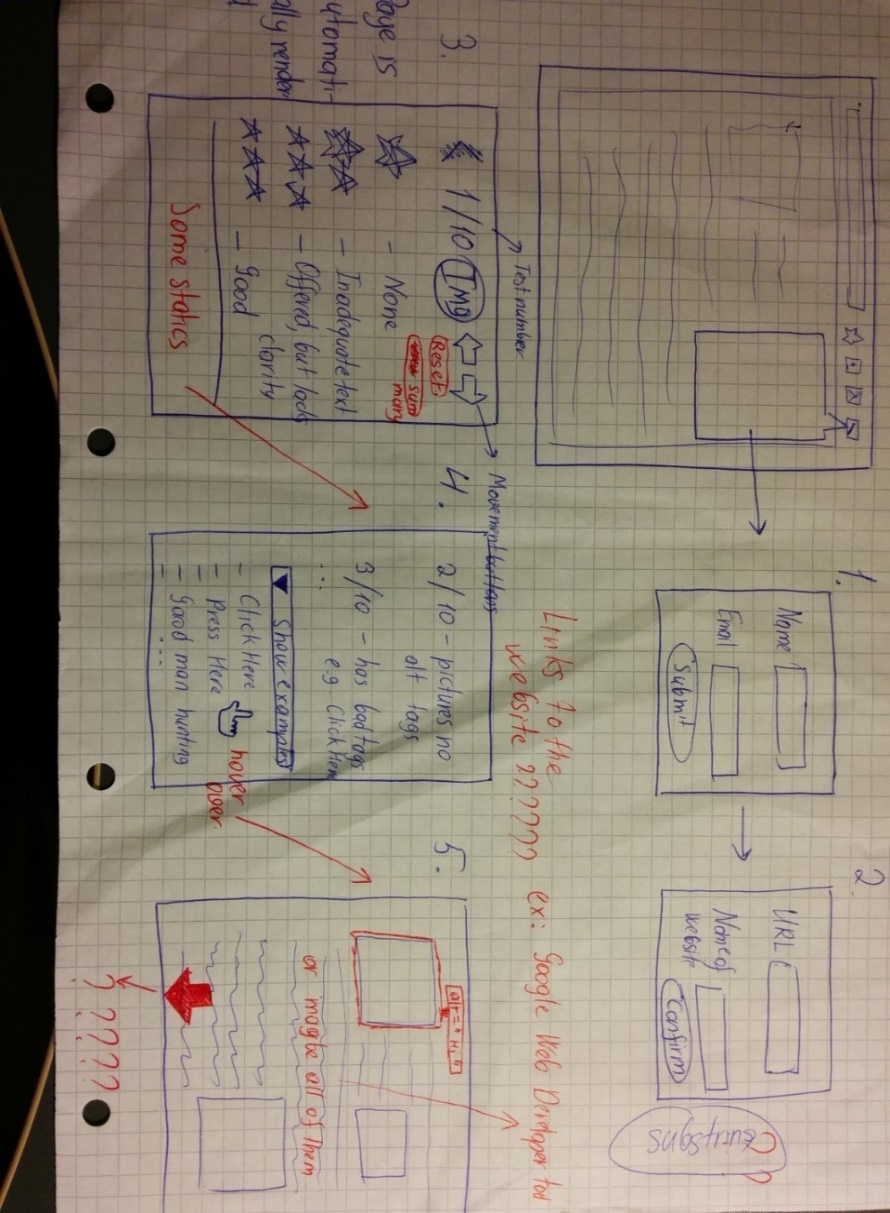
## 4.2 Client Engagement

Following the participatory spiral model approach of the project, the ECS Web Accessibility Team was met after certain milestones in order to ensure client satisfaction and document new or adjusted requirements. Short summaries of these meetings can be found in **Table 10.** The three meetings during the development process are discussed separately in **Sections** **4.3–4.6: Iterative Implementation.** They contain important explanations of solutions that were reached with regard to requirements and problems.

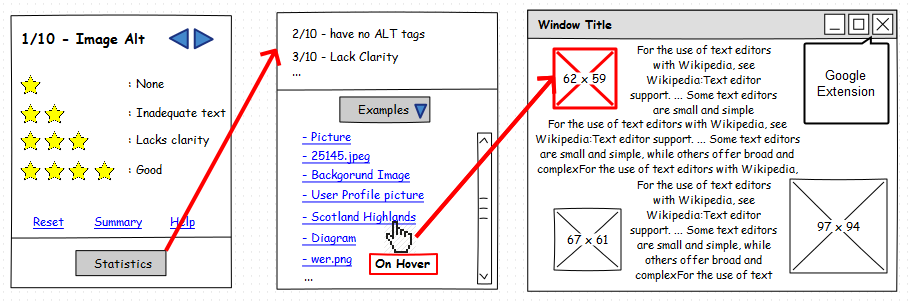
|  |  |  |  |
| --- | --- | --- | --- |
| ID | Date | Name | Summary |
| M1 | **29/09/2014** | First meeting for the design and requirements gathering | The developer brought storyboards (see **Figure 9**) to create ‘what if’ scenarios and discussion of the topic. Wrote down the list of initial requirements (see **Table 6**)**.** |
| M2 | **10/11/2014** | Second meeting for the design and requirements gathering | The developer brought wireframes with variations in design (see **Figure 10**) and wrote down additional requirements(see **Table 6**)**.** |
| M3 | **05/12/2014** | Third meeting for the design and requirements gathering | The developer brought a GE prototype (see **Figure 11**), and discussed the tool functionalities with the client (see **Table 6**). |
| M4 | **09/02/2015** | First development iteration feedback | Brought the progress of the implemented initial requirements gathered from first three meetings. Took suggestions and additional requirements(see **Table 12**). |
| M5 | **03/03/2015** | Second development iteration feedback | Brought the progress of the implemented requirements gathered from first four meetings. Fixed problems from first iteration. Took suggestions and additional requirements (see **Table 14**). |
| M5 | **20/03/2015** | Third development iteration feedback | Brought progress of implemented requirements gathered from first five meetings. Fixed problems from second iteration. Took suggestions and additional requirements (see **Table 16**)**.** |
| M7 | **02/04/2015** | Beta testing | After finalising the development, the tool was taken to the field to be tested with the client(see **Table 18**). |

**Table 10: Formal Meetings with the Client**

Using the initial knowledge about the Web2Access, storyboards were created to represent the user interface of the system. This is a common technique used in human-computer interaction and is efficient in a participatory design process (Millard et al., 2009). The rest of the storyboards can be found in **Appendix D.**



**Figure 9: Initial Storyboards**



**Figure 10: Final Wireframes**



**Figure 11: Google Extension First Prototype**

Although various kinds of functionalities were implemented simultaneously during development, the project’s development iterations could be summarised in three main categories:

* **First Iteration—**Architecture and communication
* **Second Iteration—**Main functionality and algorithms behind thetests
* **Third Iteration—**User interface, design, and code *cleaning*

## 4.3 First Development Iteration

The first implementation iteration began on February 2, 2015, and finished on February 13, 2015, and was followed by a feedback session with the client, the ECS Web Accessibility Team. Further suggestions were proposed to be included or adjusted in the overall project.

### 4.3.1 Implementation

The work during the first iteration primarily consisted of establishing solid communication between the GE and the active webpage as well as providing a clean framework for each test to be filled in at a later time. When an action, such as highlighting web elements, was performed within the GE (popup.js andpopup.html), a message was sent to the content.js file.It was embedded into the page when the GE first ran and had a listener*,* which was waiting for any upcoming messages. The GE architecture was explained in detail in **Section 3.10.4.** Using *AngularJS data-binding* functionality, in addition to the data that were displayed regarding each of the tests, the results of a user’s test marks as well as functions were stored in a global $scope variable, which can be accessed from anywhere within the application. The iteration progress is summarized in **Table 11**, which contains part of the requirements progress. A complete list can be found in **Appendix E**.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Progress Bar | Comments |
| FR1 | **5 Tests** | 34% | Average progress of all tests |
| FR1.1 | **Test 1: Forms** | 10% | Highlight all form elements and input fields |
| FR1.2 | **Test 2: Images** | 30% | Highlight all images and print their details |
| FR1.3 | **Test 3: Links** | 20% | Highlight all links and print their details |
| FR1.4 | **Test 4: CSS** | 100% | Done. Simple toggle disableproperty on <link> HTML tags |
| FR1.5 | **Test 5: Tables** | 10% | Highlight all tables |
| FR2 | **Move Between Tests** | 100% | Back/Forward buttons ready |
| FR3 | **Test Results** | 0% | Not yet begun |
| FR4 | **Highlight DOM** | 40% | Image title hovered in GE highlights respective image on page |
| FR5 | **Out of Sight DOM** | 0% | Not yet begun |
| FR6 | **Marks Summary** | 0% | Not yet begun |
| FR7 | **Reset Button** | 100% | Not only to reset all tests but a specific test as well |

**Table 11: First Iteration Development Progress**

### 4.3.2 Feedback Session

Although the team was generally satisfied with the overall work done so far**, Table 12** contains several requirements that were added to the existing requirements list as well as the issues highlighted. No interface design-based remarks were made because the team realised that this was the first iteration and thus was far from the finalised design.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Type | Name | Priority | Description |
| FR12 | **Additional Requirement** | **Test Criteria from DB** | **Compulsory** | Test criteria need to be added from the database instead of being statically added. |
| FR13 | **Additional Requirement** | **Text-to-Speech** | **Optional** | Ability to select text on the page and convert it to speech |
| FR14 | **Additional Requirement** | **Tool Accessibility** | **Compulsory** | In the tool and/or report, it has to be mentioned that the tool itself is not entirely accessible for disabled people. |
| PF1 | **Problem Fix** | **Test 3: Links** | **Compulsory** | Check for *broken* and *duplicated* links on the webpages was not implemented. |

**Table 12: First Additional Requirements/Adjustments to the Tool**

**PF1** was solved by sending an http.send() request to the link specified and returning its corresponding status (e.g., 404 or 200). This is how broken links were found.

At the end of the first iteration, it was clear that some tests in the tool were either *overestimated*—such as Test 4, ‘Removal of Stylesheet’, which took a few lines of code—or *underestimated*—such as Tests 1 and 5. Moreover, unanticipated technical difficulties have occurred during the development, such as loss of the stored data gathered from a user (i.e., scores and summaries) when the GE is closed.

## 4.4 Second Development Iteration

The second implementation iteration began on February 20, 2015, and finished on March 6, 2015, and was followed by a feedback session with the client, the ECS Web Accessibility Team. A number of problems were highlighted within the tool, and suggestions for the next development iterations were shared.

### 4.4.1 Implementation

The main focus during this iteration was to finish all the five tests presented within the tool. The tests were performed by following the list of HTML accessibility problems highlighted in WebAIM[[7]](#footnote-7) accessibility guidelines for each of the test types. The algorithm was entirely built on jQuery, whose selection methods and other functions allowed quick and efficient access to all elements of a particular type on a webpage. An example of a test’s algorithm and code can be found in **Appendix F**.

By the end of second iteration, most functionality of the five tests was finished. Moreover, the summary of progress in tests could be provided as well as a feature to view the element when its respective element was hovered over in the GE. **Table 13** shows the progress; the full list is presented in **Appendix G.**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Progress Bar | Comments |
| FR1 | **5 Tests** | 98% | Average progress of all tests |
| FR1.1 | **Test 1: Forms** | 100% | Done, using WebAIM website guidelines |
| FR1.2 | **Test 2: Images** | 100% | Done, using WebAIM website guidelines |
| FR1.3 | **Test 3: Links** | 100% | Done, using WebAIM website guidelines |
| FR1.4 | **Test 4: CSS** | 100% | Done, using WebAIM website guidelines |
| FR1.5 | **Test 5: Tables** | 90% | To be finished by solving **IF2** |
| FR6 | **Marks Summary** | 0% | Not yet begun |
| FR9 | **STEAMReader** | 0% | Left for future development, outside the project scope |
| FR11 | **Tabbing** | 50% | Being supported constantly |
| FR13 | **Text-to-Speech** | 0% | Not yet begun |
| NF1 | **Short Criteria** | 0% | Will be replaced by **FR17** |
| NF3 | **Colour Scheme** | 50% | Being supported constantly |

**Table 13: Second Iteration Development Progress**

### 4.4.2 Feedback Session

The ECS Web Accessibility Team was satisfied with the tool in general but highlighted a few technical and design problems found within the system. Furthermore, a few additional requirements emerged and were added to the existing requirements list.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Type | Name | Priority | Description |
| FR15 | **Additional Requirement** | **Navigation** | **Compulsory** | Provide easy access to tests and their summary. |
| FR16 | **Additional Requirement** | **Dynamic Content** | **Optional** | Check whether a tool works for dynamic content such as popping up a box on a click. |
| FR17 | **Additional Requirement** | **Test Criteria** | **Compulsory** | Change the wording within each criterion/ score to make it more understandable. |
| PF2 | **Problem Fix** | **Test 5: Tables** | **Compulsory** | Table test had missing ifstatement in the algorithm. |
| PF3 | **Problem Fix** | **Test Results** | **Compulsory** | Test results essential for marking are too low, and the user might not see them. |
| PF4 | **Problem Fix** | **Test Results** | **Compulsory** | Currently test results are shown based on colour match (e.g., red for ‘failed’), which is inaccessible for colour-blind people. |

**Table 14: Second Additional Requirements/Adjustments to the Tool**

As shown in **Table 14**, all additional requirements have been implemented along with the problems resolved before the commencement of the final iteration. The most time consuming problem was **PF3,** where a minimised marking box had to be created such that it stayed with the user while scrolling, i.e., fixed (**Figure 13**). **FR16,** as justified later to the client, was not implemented due to the time constraint of the project. The accessibility problems with dynamic content on webpages is currently a researched area in its own right and is beyond the scope of this project.

## 4.5 Third Development Iteration

The third and final implementation iteration began on March 3, 2015, and was finished on March 17, 2015, and was followed by a feedback session with the client, the ECS Web Accessibility Team.

### 4.5.1 Implementation

Most of the final iteration time was spent implementing additional requirements, improving the user interface, and ‘*cleaning*’ the code. *Cleaning* the code included providing comments, deleting redundant code and duplications, and separating the tests from one another in entirely independent blocks so that they could be added or removed from the GE anytime without worrying about the back-end architecture. Moreover, optional requirements such as a Text-to-Speech (TTS) converter were added to the tool in order to provide a Screen Reader functionality for testing purposes. This was achieved by using chrome.tts*,* which has been embedded into Google browsers since *Chrome 14*[[8]](#footnote-8). Finally, a summary of progress for the tests was provided as well as validation that prevents people from sending empty answers to the server. The reached progress is summarized in **Table 15**; the full list can be found in **Appendix H.**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Progress Bar | Comments |
| FR1 | **5 Tests** | 100% | Finalised all tests |
| FR3 | **Test Results** | 100% | Presented on page 6 |
| FR4 | **Highlight DOM** | 100% | Done during the second iteration |
| FR5 | **Out of Sight DOM** | 80% | Will be replaced by **FR18** |
| FR7 | **Reset Button** | 100% | Done during the first iteration |
| FR8 | **Link to Web2Access** | 100% | Done during the first iteration |
| FR11 | **Tabbing** | 100% | Done during the second iteration |
| FR13 | **Text-to-Speech** | 100% | Done using chrome.tts (Text-to-Speech) function |
| FR15 | **Navigation** | 100% | Also shows where the user is in terms of test location |
| NF4 | **Tool Accessibility** | 80% | Object highlight does not work |
| NF5 | **Speed** | 100% | AngularJS allows dynamic content change |
| NF6 | **Adaptability** | 100% | All tests are independent from one another and placed in blocks |

**Table 15: Third Iteration Development Progress**

### 4.5.2 Feedback Session

Similar to the feedback received after the second iteration, two new requirements were added to the project description, and two critical problems were highlighted within the tool. The summary of the meeting can be found in **Table 16.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Type | Name/Where? | Priority | Description |
| FR18 | **Additional Requirement** | **OnClick Scroll** | **Compulsory** | Not only hover over and highlight the element but also scroll to it on a mouse click |
| FR19 | **Additional Requirement** | **Results Summary** | **Optional** | Provide an option to view the summary of the test results that have been selected and print them |
| PF5 | **Problem Fix** | **Test 4: CSS** | **Optional** | When moving from Test 4, re-enable the CSS if disabled. |
| PF6 | **Problem Fix** | **Test Criteria** | **Compulsory** | Change the wording within each criterion/score to make it more understandable and shorter |

**Table 16: Third Additional Requirements/Adjustments to the Tool**

**FR18** and **FR19** were easily solved by using existing functions from jQuery and JavaScript, scrollTop()[[9]](#footnote-9) and window.print()[[10]](#footnote-10)*,* respectively. **PF5,** however, was much easier to fix with the simple Boolean variable update. For **PF6,** following the participatory approach of the project development, the client had sufficient experience in criteria wording to volunteer to adjust the criteria texts to their needs (see **Appendix I**).

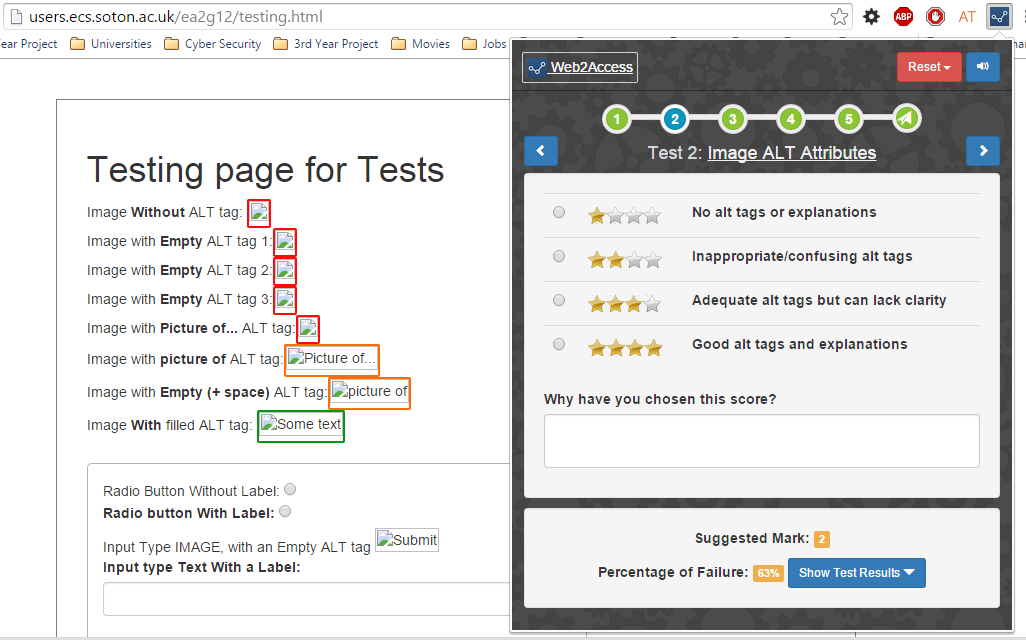
## 4.6 Final Result

The iterative participatoryspiral approach to the analysis, design, and implementation parts of this project has proved to be valuable. It allowed the team to highlight problems and implement additional requirements early in the project. This led to reduced stress, time, and consequences of late changes in the project requirements. The changing numbers of requirements and problems throughout the project development are summarised in **Table 17**.

|  |  |
| --- | --- |
| Description | Amount |
| No. of requirements gathered at the first meeting | 13 |
| No. of requirements gathered by the end of third/final meeting | 18 |
| No. of requirements gathered by the end of third/final iteration | 26 |
| No. of problems found after by the end of third iteration and beta testing (see Section 5.3) | 13 |

**Table 17: Final Number of Requirements and Problems Found**

The complete list of all functional and non-functional requirements can be found in **Appendix B.** Finally, **Figure 12** and **Figure 13** represent the final version of the GE. The rest of the screenshots can be found in **Appendix J.**



Percentage of Failure and Suggested Mark

Test-specific marking criteria and supporting reasoning

Tests, also linked to Web2Access respective pages

Navigation arrows

Text-to-Speech

Respective page elements are highlighted when navigated to the test (e.g., images)

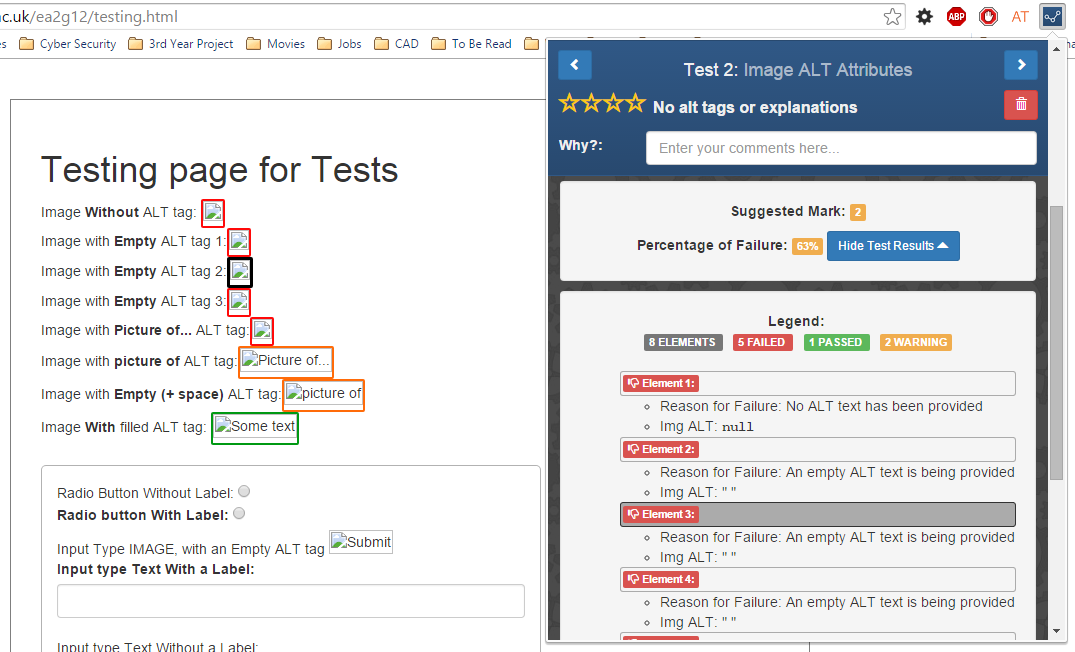
Link to Web2Access.org.uk

Navigation

Reset **the** test / **all** tests

**Figure 12: Google Extension Prototype, Screenshot 1**

Toggle button for test results details



**On hover**, element will be highlighted. **On click**, it will be scrolled to.

Test results per element

Test results summary

**Minimised** version of Marking System. Fixed.

**Figure 13: Google Extension Prototype, Screenshot 2**

The main functionalities of the GE have been discussed in the iterative development sections between **4.3** and **4.5** of the report. Those few additional features that weren’t discussed there are listed below:

* **Minimised** and **fixed** version of the Marking System appears only after a specific amount of pixels are scrolled down—in our case, when the main Marking System is scrolled out of sight.
* **On submission** of the results, they will be validated and checked as to whether every field has been filled in.
* **The suggestion mark** is being calculated by dividing number of failed elements to the total number of elements on the page. The resulted percentage was matched to the 4-scaled marking system, e.g., 26% = 2, 53% = 3.
* The table of content of the design archive file, i.e., project’s main folder, can be found in **Appendix Q.**

# 5. Testing and Evaluation

## 5.1 Testing Strategy

Having been built on AngularJS, the GE is a one-page application, which means that there are no heavy user interface functionalities; there are only Reset and Submit buttons that must be tested. The key elements of the GE are its web accessibility tests, which were the main target during the testing phase. The primary goal of the testing process was to identify whether the tool has achieved one of the main research goals of the project:

* **No. 3: Tool specificity and completeness**

In other words, do the tests highlight all possible inaccessible code types? For this to be checked, a testing page[[11]](#footnote-11) was created on the developer’s public server with a variety of deliberate HTML errors. Results from the GE were compared to the results for the same page with WAVE web accessibility checker. WAVE[[12]](#footnote-12) is the tool most commonly used across the Web2Access accessibility tests.

## 5.2 Test Results

Deliberate errors within the testing page were created following the guidelines from WebAIM for each accessibility test. Results from these tests have been summarised in **Table 18**; the full list can be found in **Appendix K.**

|  |  |  |  |
| --- | --- | --- | --- |
| Accessibility Test | ID | Description | Test Result |
| Test 1: Forms | | | |
|  | **1.** | Radio button with/without label | Passed |
|  | **2.** | Input type image, with/without empty alt tag | Passed |
|  | **3.** | Input type text, with/without a label | Passed |
|  | **4.** | Textarea with/without a label | Passed |
| Test 2: Images | | | |
|  | **1.** | Image without alt tag | Passed |
|  | **2.** | Image with empty alt tag | Passed |
|  | **3.** | Image with empty (+ space) alt tag | Passed |
|  | **4.** | Image with filled alt tag | Passed |
| Test 3: Links | | | |
|  | **1.** | Link without a href attribute | Passed |
|  | **2.** | Link with empty href attribute | Passed |
|  | **3.** | Link with empty ( + spaces) attribute | Passed |
|  | **4.** | Link with empty text | Passed |
| Test 4: CSS |  | | |
|  | **1.** | Enable/disable CSS button toggle | Passed |
| Test 5: Tables | | | |
|  | **1.** | A table with <thead> and <th> | Passed |
|  | **2.** | A table with <thead> but without <th> | Passed |
|  | **3.** | A table with <th> but without <thead> | Passed |
|  | **4.** | A table without <thead> or <th> | Passed |

**Table 18: Test Cases and Results**

During the testing process, it was additionally found that in some cases the GE highlighted more errors than the WAVE accessibility checker did. For examples see **Figure 14.**



**Figure 14: Comparison between GE (left) and WAVE (right)**

## 5.3 Beta Testing

After performing the test cases, it was decided to do *beta testing* in *the field* with the client. Instead of using the developer’s computer to test the deliberately created testing page, the client was sent the GE in .rar format and was asked to install it himself. He was using his own PC and could choose any website for testing. Advantages of the field testing, such as non-intrusiveness, comfort, and familiarity with one’s own device (Yvonne et al., 2007), allowed more focus on the tool and helped the user find yet unknown problems within the tool. These problems are summarised in **Table 19.**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Where? | Problem | Fixed By |
| PF7 | **Test 3: Links** | URL links are white colour, which makes them unreadable. | Add a separate CSS class and adjust it |
| PF8 | **Test 3: Links** | Links without text in them are hard to be found. | By implementing **FR18:** onclick =”scrollTop()”[[13]](#footnote-13) |
| PF9 | **Summary** | Tests on summary page are labelled incorrectly: ‘Test 1’, ‘Test 1’, ‘Test 1’, etc. | Minor JavaScript problem |
| PF10 | **Reset Button** | Text in submenus is white, which makes it unreadable. | Add a separate CSS class and adjust it |
| PF11 | **Test 4: CSS** | When disabling CSS on the pages, it also disables CSS in the Reset button, which makes its submenus inaccessible. | Removing AngularJS variable duplication within the code |
| PF12 | **Progress Bar** | Evaluator intuitively starts to press test numbers in the progress bar trying to navigate to a specific test. | Provide anchor tags (<a>) and link them to each test. |
| PF13 | **Architecture** | When the GE is closed, test results gathered from the user are not stored. | Use of the built-in browser storage: chrome.storage()[[14]](#footnote-14) |

**Table 19: Problems Highlighted During the Beta Testing**

## 5.4 Evaluation Goals

The main purpose of the evaluation was to answer the important question in this project: ‘*Were the Web2Access problems solved and the project goals achieved?’* This was checked by asking a specific set of questions regarding each of the project goals established at the beginning of this project:

1. **Minimize the effort required to perform the tests.**
2. **For the experienced, speed up the process; for the inexperienced, make it actually feasible.**
3. **Ensure tool specificity and completeness.**
4. **Secure more participants with Web2Access.**
5. **Balance between automatic and manual work.**

Additional evaluation goals were to find functionality and usability problems with the tool that had not been discovered during the beta testing with the client and to seek any ideas that could be implemented in the future development of the tool.

## 5.5 Evaluation Approach

Because of the scope of the project, only five people were chosen to participate in the evaluation process. According to previously conducted research, it has been found that the best results come from tests with no more than *five* people (Nielsen & Landauer, 1993). All participants were part of the ECS research group and had at least some knowledge about Web2Access and web accessibility in general. These evaluators were met individually in field conditions to provide a comfortable environment. Each person was given the same set of evaluation tasks (see **Appendix L**) to cover, which allowed them to familiarise themselves with the system. Evaluators were asked to use the *think-aloud protocol* (Benbunan-Fich, 2011)so that the researcher could record their thought processes while accomplishing the tasks. During the evaluation, a few problems were mentioned regarding the functionality and user interface of the GE (see **Appendix M**).

After the evaluation tasks were completed, each evaluator was asked a set of predefined questions in a *structured interview* manner (Yvonne et al., 2007). Ethics approval for the interviews was provided by the University of Southampton (see **Appendix N**). The questions were *open* type in order to trigger a discussion around the topic. All questions were related to the research goals of the study (see **Appendix O**).

## 5.6 Evaluation Results

After the evaluation tasks were completed by the participants, it was concluded that the tool was quite intuitive and easy to use. Moreover, the participants managed to complete the evaluation tasks without further assistance. Both of these factors indicate that usability goals of the study for *efficiency* and *learnability* have been fulfilled. However, during the completion of the evaluation tasks, a few user interface problems were raised and are listed in **Appendix M**. The usability goal for *utility* was tested in the next part of the evaluation process. The usability goal for *maintainability,* however, was difficult to test under the given conditions.

The final part of the evaluation consisted of a set of predefined questions about the project’s goals that were asked in a structured interview manner. The summary of the achievements of the project’s goals is presented below backed up with citations from evaluators.

1. **Minimize** **the effort required to perform the test.**

The user feedback confirmed that the effort required to perform the tests has been minimized, because now the web accessibility tool, as it was commented, “*runs in the context of the page*”. Most of the evaluators mentioned the reduction of “*jumping”* between different windows on the desktop. One of the evaluators said: “*Previously I had to use three screens to perform the tests—one for Web2Access review marking system, one for a web accessibility checker I used, and one for the tested website. However, now I can do everything in one browser window using the GE*”. A few solutions to user interface problems were suggested. Additionally, the evaluators highlighted that unlike many other accessibility checkers, the GE doesn’t need the Internet connection for functioning, which makes it suitable for remote development. Moreover, the WAVE and other accessibility tools struggle with the pages that require authentication to proceed further (“*I can finally Login*”), due to Cross-Site Scripting permissions, when the GE doesn’t.

1. **For the experienced, speed up the process; for the inexperienced, make it actually feasible.**

Evaluators who were **experienced** in web accessibility have estimated that the time required to perform these tests was reduced by at least *half,* which confirms that the speedup goal has been achieved. Example comments are: “*It took me 20-30 minutes to do a review previously, with the GE, however, it would take me half of that time*”. One of the evaluators suggested the placement of a *loading bar* for time-consuming tests, and the other evaluator said that it is wasteful to show *passed* web elements in the GE and to instead only show the *failed* ones.

Responses from **inexperienced** evaluators, however, varied. Although the majority said that the tool was “*easy to use”*, two evaluators mentioned that it was not intuitive to use the tool at the beginning. A proposal was made to provide a short five-minute tutorial that covers the main functionalities of the tool. This tutorial could be placed either on the Web2Access website or on the Google Web Store, where the GE is planned to be published.

1. **Ensure tool specificity and completeness.**

This project goal had one of the best responses overall. The aim was to check whether the results from the automated checker were better or worse than the tools used previously. The evaluators confirmed that the goal has been achieved. One evaluator said that the results were “*just as good*” as in WAVE, our biggest competitor. The other one said that highlighting methods of the passed/failed elements are better than in WAVE because the latter adds additional DOM elements to the website, which displaces the website element positions and “*makes me confused*” as he said, whereas the GE simply puts a border around it. The third evaluator said that it was easier for him to mark the website when all test results were in one place, in the GE, rather than being spread all over the webpage as in WAVE. He stated: “*It is better to have all errors in one place rather than checking every element one by one on the webpage*”. Suggestions for improvement were made, such as making the error messages more descriptive for novice users and showing all errors regarding a specific element if it failed in more than one accessibility category.

1. **Secure more participants with Web2Access.**

This could be the weakest point in the project with a result of two positive and three negative feedback results. One evaluator with positive feedback stated that if we advertise how useful this tool is for solving web accessibility problems and the necessity of following the accessibility guidelines, it should help us increase the number of participants. The other evaluator stated that: “*this tool works in a similar manner to that of the auto-fix feature in the Visual Studio text editor*”. The difference is that in Visual Studio, accessibility errors are highlighted when written (e.g., an image requires an ALT tag), but the GE will highlight the errors in a browser that the developer will have to use because he is developing for the web. The three evaluators with negative feedback considered themselves as being motivated to perform more checks because the GE makes their current lives easier. However, they all mentioned that they “*do not see how this tool could add motivation for others*”, who are not already participating to join the Web2Access review process.

1. **Balance between automation and manual work.**

This project goal emerged from the background research on the Web2Access. It was concluded that the severity score can be quite subjective when it comes to choosing a mark between, for instance, 2 and 3. Four out of five participants agreed that having a ‘*suggestion*’ system to assist in calculating the marking would be a good idea. Some evaluators substantiated their choice by saying that it will be “*easier for me to decide between close numbers*”. Some asserted that it would simplify the marking if there were so many elements on a page that it would be “*hard and time taking to calculate percentage of failed elements*” (e.g., 573 images on the page, 193 of which failed). However, one evaluator had a reasonable point against this *suggestion* system based on the fact that as he stated “*the suggestion could bias the answers of the users*”. As he explained, it is difficult to completely automate any check on a webpage, and the tool may miss some advanced problems on the page, thus making the *suggestion* unrepresentative of reality and hence lead the user in the wrong direction. Although in the minority, this point highlighted by the evaluator requires further research.

## 5.7 Future Evaluation

When this tool is extended and finalised in the future, a larger number of evaluators must be added to the study. If more people participate, *quantitative* data would be more sensible to collect, because the results would be more realistic. Additionally, the study should include the disabled people themselves who will benefit from the product. In the end, they are the ones whose lives this tool is intended to make easier. Disabled people were not asked to participate in the current study mainly because of the time constraints and possible problems with ethics approval by the university, which could lead to delays in progress.

Additionally, the evaluators should be divided into two separate groups. One group should perform the Web2Access tests in the old manner with the previous checks and the other group would use only the GE. Both of the results would be compared, and if the results from the GE were to differ significantly from the previous methods, then the tool would require further study. The tool should not affect the results of the web testing; it is meant to speed up the process and make the evaluator’s life easier.

# 6. Conclusion

## 6.1 Project Goals Evaluation

The main goal of providing a working prototype to help users perform five web accessibility tests using the Web2Access service has been achieved by fulfilling nearly all compulsory requirements provided by the client. The project goals in particular, however, had some variations (see **Section 5.6: Evaluation Results** for the evidences):

1. **Minimize the effort required to perform the tests:** The tool has successfully managed to reduce the numbers of screens/windows being used and various tools being downloaded and installed.
2. **For the experienced, speed up the process: for the inexperienced, make it actually feasible:** The process was sped up by at least double. However, they also thought that the system was not sufficiently intuitive for a novice user and that the descriptions lack satisfactory details.
3. **Ensure tool specificity and completeness:** Simplifying the tool did not affect its functionality, specificity, or completeness compared to other web Accessibility tools, in some places even surpassing them.
4. **Secure more participants with Web2Access:** Most of the evaluators (three out of five) did not see how this tool could encourage new participants to download and use it.
5. **Balance between automatic and manual work:** The proposed suggestion system for the marking was mostly welcomed with the caveat that it be researched further.

Finally, it would have been a good idea to separate the evaluation of the user interface from evaluation targeting the research and project goals. This would have allowed better focus on each of the areas.

## 6.2 Participatory Spiral Model

Using a spiral model for the software development lifecycle was successful because it allowed instant adaptation to the client’s new requirements and adjustments. The benefits of the iterative approach in the spiral model were mostly significant during requirements gathering/analysis, design, and implementation. Information from previous phases was used as a basis for discussion and development in the next ones. As a result of the regular iterations and meetings, the number of the client’s requirements doubled from 13 to 26, thus bringing up needs that otherwise would have remained hidden. All of these results led to better client satisfaction at the end of the project.

## 6.3 Implementation and Tools

As mentioned, the iterative approach allowed us to fulfil all compulsory requirements in the project. A better idea for the future work, however, would be to categorise the requirements from the client so that when one requirement is fulfilled, a similar one could be solved without having to rethink the whole process from the beginning. The testing of the algorithms was also quite superficial. More websites should be tested with the tool, and the results should be compared with those from the other web accessibility checkers in a more detailed manner. The introduction of the unit testing for coding could also be a good idea. Finally, during the user evaluation, more technical problems appeared that could shift the focus of the evaluation from the main target, the research goals. The suggestion would be to perform at least three beta tests in the field instead of one to maximize the number of solved technical problems.

The AngularJS framework was easy to learn and use during the project. Its ‘data-binding’ between the variables was particularly useful. Its code, however, must be embedded into the normal HTML, which makes the code somewhat harder to maintain by other developers. However, it was a necessary tool and resulted in a simple single-page application. Additionally, the jQuery library was irreplaceable in finding and highlighting the HTML elements according to web accessibility guidelines, but it did require a pedantic and carefully thought algorithm. Bootstrap saved time on further CSS adjustments in the user interface.

## 6.4 Schedule

The project was delivered mostly within the time scale estimated at the beginning. However, some additional time for implementation would have allowed delivery of better-quality code. Furthermore, the ethics form for the tool evaluation was rejected by the university six times, which caused a delay in the project and forced the evaluation to be conducted over the Easter break. This increased the difficulty of accessing the evaluators over the holidays. In the future, more flexibility in time should be allowed for the tasks that do not depend on the developer himself, such as ethics approval from the university.

## 6.5 Future Work

The features mentioned here are mainly taken from the optional requirements that have not been implemented during the project. Moreover, it contains suggestions gathered from evaluators and users proposed during beta testing, evaluation processes, and iterative meetings as well as the developer’s personal thoughts on possible future areas for development. The summary of the suggestions is presented in **Table 20.**

|  |  |  |
| --- | --- | --- |
| ID | Suggestion | Comments |
| 1. | **Finish All 15 Tests** | Provide additional 10 accessibility tests from Web2Access to already developed five tests. |
| 2. | **Finish the GET and POST APIs** | GET API will be used to get data regarding tests (e.g., text, name, id). POST API will receive test results from the GE on users’ submit. |
| 3. | **Gamification** | To achieve one of the research goals and increase the number of participators the tool should include some gamification elements, e.g., badges/points for success |
| 4. | **Tablet Size** | Another suggestion was to provide a tablet version of the tool to allow users to work even on mobile devices that have Google Chrome browser being installed. |
| 5. | **STEAMReader** | The optional requirement **FR9,** add STEMReader for checking the MathML content of the website. |
| 6. | **User Contact Details** | To be stored in DB and be associated with the test results the system needs to have a user details to link them in database. |
| 7. | **Evaluation Problems** | Fulfil most of the suggestions and solve the problems mentioned during the evaluation process with the users, **Appendix M.** |
| 8. | **Dynamic Content** | GE has to be improved to react on the dynamic content as well such as popping up boxes or sliding pictures |
| 9. | **Secure Insertion** | The server connection must be secured so no SQL injections could be sent over POST API. |
| 10. | **Other Browsers** | After fully developed it has to be considered to extend the extension to other browsers, such as Mozilla Firefox. |

**Table 20: Future Work**

## 6.6 Conclusion

A prototype was developed that has addressed the main problems of performing web accessibility tests on the Web2Access service. The tool was delivered on time and has fulfilled all compulsory requirements established by the client. A participatory spiral model was used as the software development lifecycle, which helped easily accommodate emerging and changing requirements during the project’s development. The client was highly satisfied with the delivered product and the developer was offered an opportunity to finalise the tool as an intern in the ECS research group over the summer 2015. The main focus for the future development should be the automation of the other ten tests to facilitate faster and easier user decision making.

# References

Bailey, J., & Burd, E. (2006). What is the current state of Web Accessibility? *Web Site Evolution* (pp. 69-74). Philadelphia: IEEE.

Benbunan-Fich, R. (2011). Using protocol analysis to evaluate the usability of a commercial web site. *Information & Management, 39*(2), 151-163.

Boehm, B. W. (1988). A spiral model for software development and enhancement. *IEEE Computer*, 61-72.

Boehm, B., & Bose, P. (1994). A collaborative spiral software process model based on Theory W. *Applying the Software Process* (pp. 59-68). Reston: IEEE.

Brajnik, G., Yesilada, Y., & Harper, S. (2010). Testability and Validity of WCAG 2.0: The Expertise Effect. *International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 43-50). Orlando: ACM.

Brajnik, G., Yesilada, Y., & Harper, S. (2011). The Expertise Effect on Web Accessibility Evaluation Methods. *Human-Computer Interaction*, 246-283.

Brewer, J. (2004). Web Accessibility Highlights and Trends. *World Wide Web Coneference* (pp. 51-56). New York: ACM.

Brophy, P., & Vraven, J. (2007). Web Accessibility. *Library Trends*, 950-972.

Bryson, J. M. (2007). What to do when Stakeholders matter. *Public Management Review*, 21-53.

Craven, J. (2006). Web Accessibility: a review of research and initiatives. *72ND IFLA GENERAL CONFERENCE AND COUNCIL*, (pp. 1-11). Seoul.

Dobransky, K., & Hargittai, E. (2006). The Disability Divide in Internet Acces and Use. *Information, Communication and Society* , 313-334.

Fisher, C. (2010). Getting plugged in: An overview of Internet addiction. *Paediatrics and Child Heath*, 557-559.

Hansen, T. R. (2004). Socio-Technical Experiments – a New Approach on the Design Process. *Electronic Colloquium on Computational ...*, 1-10.

Henry, S. L., Abou-Zahra, S., & Brewer, J. (2014). The Role of Accessibility in a Universal Web. *11th Web for All Conference* (pp. 1-4). Seoul: ACM.

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, 120-127.

Ivory, M. Y., Mankoff, J., & Le, A. (2003). Using Automated Tools to Improve Web Site Usage by Users with Diverse Abilities. *Information Technology and Society*, 1-46.

Karim, R., Dhawan, M., & Ganapathy, V. (2014). Retargetting Legacy Browser Extensions to Modern Extension Frameworks. *ECOOP Object-Oriented Programming* (pp. 463-488). Uppsala: Springer Berlin Heidelberg.

Kasday, L. R. (2000). A Tool to Evaluate Universal Web Accessibility. *CUU '00 Proceedings on the conference on Universal Usability* (pp. 161-162). Philadelphia: ACM.

Lazar, J., Dudley-Sponaugle, A., & Greenidge, K.-D. (2004). Improving web accessibility: a study of webmaster perceptions. *Computers in Human Behavior*, 269-288.

Leporini, B., Paterno, F., & Scorcia, A. (2006). Flexible tool support for accessibility evaluation. *Interacting with Computers*, 869-890.

Maeda, J., Fukuda, K., Takagi, H., & Asakawa, C. (2004). Web Accessibility Technology at the IBM Tokyo Research Laboratory. *Research and Development*, 735-749.

Millard, D., Howard, Y., Gilbert, L., & Wills, G. (2009). Co-design and Co-deployment Methodologies for innovative m-Learning Systems. 1-14.

Mujumdar, A., Masiwal, G., & Chawan, P. M. (2012). Analysis of various Software Process Models. *International Journal of Engineering Research and Applications (IJERA)*, 2015-2021.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Adaptive complex enterprises*, 72-78.

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Conference on Human Factors in Computing Systems* (pp. 206-213). New York: ACM.

Palmieri, M., Signh, I., & Cicchetti, A. (2012). Comparison of Cross-Platform Mobile Development Tool. *Intelligence in Next Generation Networks (ICIN)* (pp. 179-186). Berlin: IEEE.

Pascual, A., Ribera, M., Granollers, T., & Coiduras, J. L. (2013). Impact of accessibility barriers on the mood of blind, low-vision and sighted users. *International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion* (pp. 431-440). Barcelona: Procedia Computer Science.

Richards, G., Lebresne, S., Burg, B., & Vitek, J. (2010). An Analysis of the Dynamic Behavior of JavaScript Programs. *SIGPLAN conference on Programming language design and implementation* (pp. 1-12). New York: ACM.

Say-Wei, F., & Muruganantham, A. (2000). Software risk assessment model. *Management of Innovation and Technology* (pp. 536-544). New-York: IEEE.

Shaked, U. (2014, August 16). *AngularJS vs. Backbone.js vs. Ember.js*. Retrieved from AirPair: https://www.airpair.com/js/javascript-framework-comparison

Trewin, S., Cragun, B., Swart, C., Brezin, J., & Richards, J. (2010). Accessibility Challenges and Tool Features: An IBM Web Developer Perspective. *19th International World Wide Web Conference* (pp. 1-10). USA: ACM.

W3Counter. (2015, March 30). *February 2015 Web Browser Market Share*. Retrieved from Global Stats: http://www.w3counter.com/globalstats.php

Wald, M., Draffan, E., Newman, R., & Skuse, S. (2012). Access Toolkit for Education. *13th International Conference Computers Helping People with Special Needs* (pp. 51-58). Linz: Springer.

Wald, M., Draffan, E., Skuse, S., Newman, R., & Phethean, C. (2011). Southampton Accessibility Tools. *20th International World Wide Web Conference* (pp. 1-2). Hyderabad: ACM.

Xiong, J., Farenc, C., & Winckler, M. (2007). Analyzing Tool Support for Inspecting Accessibility Guidelines During the Development Process of Web Sites. *Web Information Systems Engineering – WISE Workshops* (pp. 470-480). Berlin: Springer.

Yvonne, R., Jenny, P., & Sharp, H. (2007). *Interaction Design: beyond human-computer interaction* (2nd ed.). Chichester: John Wiley & Sons, Ltd.

# Appendices

## Appendix A: Primary Stakeholders – Personas and Scenarios

|  |  |
| --- | --- |
| Persona | Description |
| C:\Users\ELDAR\Desktop\98f69e4e91cbd09480155c2b10e0707b.jpg | Russell is currently a postgraduate in full-time research in Web and Internet Science. Studying both his Bachelors and Masters at the University of Southampton he has been involved in several accessibility projects, such as LexDis, Web2Access and ATbar. Moreover, Web2Access was built as a part of his summer internship during his studies. |
| **Scenario**  Apart from research within the department Russell also owns his own IT company which is mostly specialized in web services development. AS a former researcher Russell is interesting in providing accessibility over their websites. However, problem that most employees face is the amount of tools required to be downloaded or used and their variety |

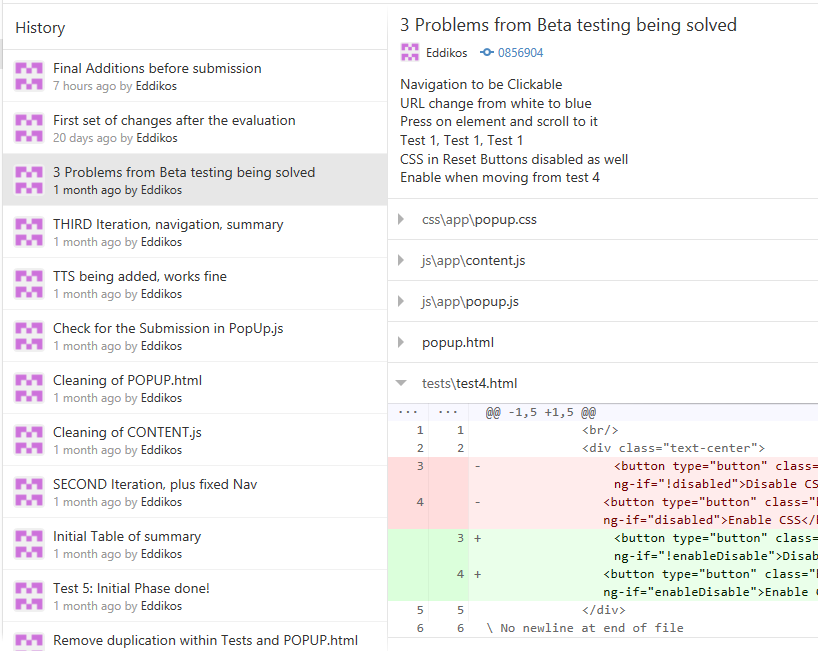
|  |  |
| --- | --- |
| Person | Description |
|  | Lucy is a housewife and mother of two teenagers. Family lives in the suburbs of London, UK, one of her children has visual disabilities, in terms of colour blindness, since he was born. However, he is still able to use all learning facilities provided in the school, but with a help of some accessibility tools. |
| **Scenario**  Currently Lucy is using free online tools in order to check the accessibility of the online tools that her son is currently using to find the most suitable/accessible ones. Even though results from some tools are quite detailed and very useful for her, their technical content doesn’t allow Lucy use them effectively |

## Appendix B: Final List of Requirements

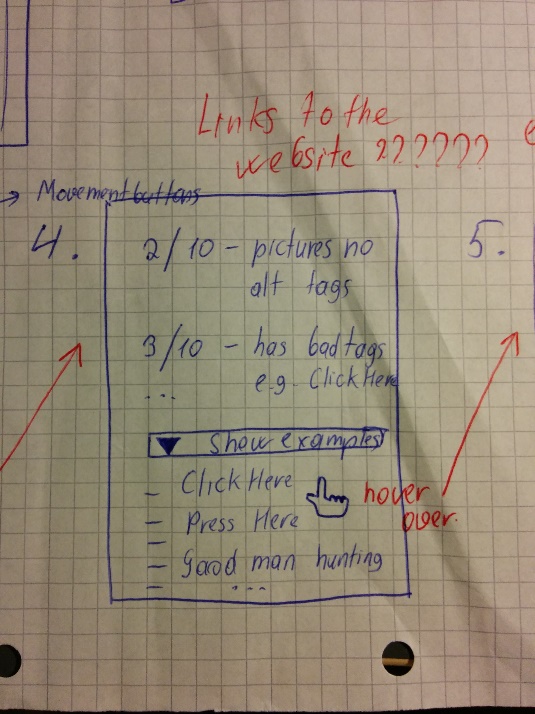
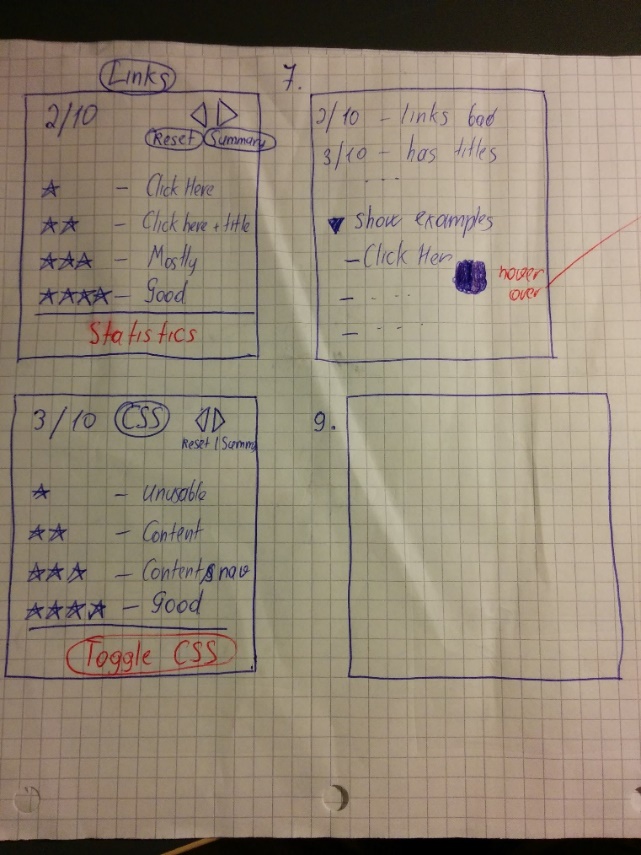
* First Meeting with the Client
* Second Meeting with the Client
* Third Meeting with the Client
* First Development Iteration
* Second Development Iteration
* Third Development Iteration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Requirements and Description | Priority | Done |
| FR1 | **5 Tests** | Provide at least five various checks/tests fully working and being able to communicate with the server | **Compulsory** | **Yes** |
| FR2 | **Move between tests** | Ability to easily move between checks, e.g. back and forward arrows | **Compulsory** | **Yes** |
| FR3 | **Test Results** | A way to view Statistics and Examples of performed test results. | **Compulsory** | **Yes** |
| FR4 | **Highlight DOM** | For DOM (Document Object Model) related checks, allow to view/ highlight the specific element on the page by hovering it from the extension | **Compulsory** | **Yes** |
| FR5 | **Out of Sight DOM** | Show with arrows on the page if the DOM element is outside of the browser window and user need to scroll down | **Optional** | **Yes** |
| FR6 | **Marks Summary** | View the Summary of progress for the checks on a specific website. | **Compulsory** | **Yes** |
| FR7 | **Reset Button** | To delete the test or all tests | **Compulsory** | **Yes** |
| FR8 | **Link to Web2Access** | Provide Links to the main website (Web2Access), either for reference purpose or for more detailed information | **Compulsory** | **Yes** |
| FR9 | **STEAM-Reader** | Add STEMReader for checking the MathML content of the website | **Optional** | **No** |
| FR10 | **Contact Details** | Take user contact information, optionally | **Optional** | **No** |
| FR11 | **Tabbing** | Allow tabbing for a non-mouse access | **Compulsory** | **Partly** |
| FR12 | **Test Criteria from DB** | Test Criteria need to be added from database instead of being statically added. | **Compulsory** | **No** |
| FR13 | **Text-To-Speech** | Ability to select text on the page and convert it to speech | **Optional** | **Yes** |
| FR14 | **Tool Accessibility** | In the tool and/or in the report it has to be mentioned that the Tool itself is not entirely accessible for disabled people. | **Compulsory** | **Partly** |
| FR15 | **Navigation** | Provide easy access to tests and their summary | **Compulsory** | **Yes** |
| FR16 | **Dynamic Content** | Check whether tool works for dynamic content, such as popping up box on click | **Optional** | **No** |
| FR17 | **Test Criteria** | Change the wording within each criteria/ score, to make it more understandable and shorter | **Compulsory** | **Yes** |
| FR18 | **OnClick Scroll** | Not only hover over and highlight the element, but also scroll to it on a mouse click | **Compulsory** | **Yes** |
| FR19 | **User Answers Summary** | Provide an option to view the summary of the test results you’ve selected and print them out | **Optional** | **No** |
| NF1 | **Short Criteria** | Reduce presented to user information per test/check as much as possible due to the small sizes of Google Extension. Was replaced later on with **FR14** | **Compulsory** | **Yes** |
| NF2 | **Gamification** | Badges for those who have done several tests and compare them to others | **Optional** | **No** |
| NF3 | **Colour Scheme** | Support the colour scheme presented on Web2Access website | **Compulsory** | **Yes** |
| NF4 | **Tool Accessibility** | Being an accessibility tool, it has to be accessibility itself. Was replaced later on with **FR14** | **Optional** | **Partly** |
| NF5 | **Speed** | Response time/ Performance – it must allow faster testing | **Compulsory** | **Yes** |
| NF6 | **Adaptability** | Easy integration of additional tests in future development | **Compulsory** | **Yes** |
| NF7 | **Sequence of Tests** | Easy to automate checks first and then the rest | **Optional** | **Yes** |

## Appendix C: Project’s Implementation History in GitHub



## Appendix D: Storyboards

## Appendix E: First Iteration Development Progress

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Progress Bar | Comments |
| FR1 | **5 Tests** | 34% | Average progress of all tests |
| FR1.1 | **Test 1: Forms** | 10% | Highlight all form elements and input fields |
| FR1.2 | **Test 2: Images** | 30% | Highlight all images and print their details |
| FR1.3 | **Test 3: Links** | 20% | Highlight all links and print their details |
| FR1.4 | **Test 4: CSS** | 100% | Done. Simple toggle of *disable* property on *<link>* HTML tags |
| FR1.5 | **Test 5: Tables** | 10% | Highlight all tables |
| FR2 | **Move between tests** | 100% | Back and Forward buttons ready |
| FR3 | **Test Results** | 0% | Not yet begun |
| FR4 | **Highlight DOM** | 40% | Image title hovered in Extension highlights respective image on page |
| FR5 | **Out of Sight DOM** | 0% | Not yet begun |
| FR6 | **Marks Summary** | 0% | Not yet begun |
| FR7 | **Reset Button** | 100% | Not only to reset all tests, but a specific one as well |
| FR8 | **Link to Web2Access** | 100% | The test’s name acts as a link |
| FR9 | **STEAMReader** | 0% | Being an optional feature was left to the end |
| FR10 | **Contact Details** | 0% | Still to be made decision up on |
| FR11 | **Tabbing** | 60% | Possible to tab within the tool, but not to get to it |
| NF1 | **Short Criteria.** Was replaced later on with **FR14** | 20% | Text needs to be shortened, waiting for the final adjustments in the database. |
| NF2 | **Gamification** | 0% | Gamification aspect can be added only when all tests are ready |
| NF3 | **Colour Scheme** | 0% | Styling will be mainly focused at the available time at the end of the development |
| NF4 | **Tool Accessibility**. Was replaced later on with **FR14** | 60% | Possible to tab within the tool, but not to get to it |
| NF5 | **Speed** | 60% | Being completely client-side and lightweight JavaScript provides a dynamic access |
| NF6 | **Adaptability** | 30% | Code is optimized on-go, and by separation tests are made to be architecture independent |
| NF7 | **Sequence of Tests** | 100% | Checks implemented in the exact same sequence as on the website |

## Appendix F: Example of Code and Algorithm – Test 2: Images



## Appendix G: Second Iteration Development Progress

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Progress Bar | Comments |
| FR1 | **5 Tests** | 98% | Average progress of all tests |
| FR1.1 | **Test 1: Forms** | 100% | Done, using WebAIM website guidelines |
| FR1.2 | **Test 2: Images** | 100% | Done, using WebAIM website guidelines |
| FR1.3 | **Test 3: Links** | 100% | Done, using WebAIM website guidelines |
| FR1.4 | **Test 4: CSS** | 100% | Done, using WebAIM website guidelines |
| FR1.5 | **Test 5: Tables** | 90% | To be finished by solving **IF2** |
| FR2 | **Move between tests** | 100% | Done during first iteration |
| FR3 | **Test Results** | 80% | Prepared but additional styling is required |
| FR4 | **Highlight DOM** | 100% | Done, using onmouseoverandonmouseoutevents. For all tests |
| FR5 | **Out of Sight DOM** | 0% | Not yet begun, searching for the solution |
| FR6 | **Marks Summary** | 0% | Prepared but not properly displayed yet |
| FR7 | **Reset Button** | 100% | Done during the first iteration |
| FR8 | **Link to Web2Access** | 100% | Done during the first iteration |
| FR9 | **STEAMReader** | 0% | Left for future development, outside of the project scope |
| FR10 | **Contact Details** | 0% | Left for future development, as contains personal sensitive data |
| FR11 | **Tabbing** | 50% | Being supported constantly |
| FR12 | **Test Criteria from DB** | 0% | Waiting for the help from ECS Accessibility Team, the client |
| FR13 | **Text-To-Speech** | 0% | Not yet begun |
| FR14 | **Tool Accessibility** | 60% | Being supported constantly |
| NF1 | **Short Criteria** Was replaced later on with **FR14** | 0% | Will be replaced by **FR17** |
| NF2 | **Gamification** | 0% | Left for future development, outside of the project scope |
| NF3 | **Colour Scheme** | 50% | Being supported constantly |
| NF4 | **Tool Accessibility**. Was replaced later on with **FR14** | 70% | Object highlight doesn’t work |
| NF5 | **Speed** | 80% | AngularJS allows dynamic content change |
| NF6 | **Adaptability** | 40% | Initial removal of duplicated code |
| NF7 | **Sequence of Tests** | 100% | Followed as it is in the Web2Access website |

## Appendix H: Third Iteration Development Progress

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Progress Bar | Comments |
| FR1 | **5 Tests** | 100% | Finalised all tests |
| FR2 | **Move between tests** | 100% | Done during the first iteration |
| FR3 | **Test Results** | 100% | Main test results are presented |
| FR4 | **Highlight DOM** | 100% | Done during the second iteration |
| FR5 | **Out of Sight DOM** | 80% | Will be replaced by **FR18** |
| FR6 | **Marks Summary** | 100% | Presented on the page 6 |
| FR7 | **Reset Button** | 100% | Done during the first iteration |
| FR8 | **Link to Web2Access** | 100% | Done during the first iteration |
| FR9 | **STEAMReader** | 0% | Left for future development, outside of the project scope |
| FR10 | **Contact Details** | 0% | Left for future development, as contains personal sensitive data |
| FR11 | **Tabbing** | 100% | Done during the second iteration |
| FR12 | **Test Criteria from DB** | 0% | Waiting for the help from ECS Accessibility Team, the client |
| FR13 | **Text-To-Speech** | 100% | Done using Chrome.tts (Text-To-Speech) function |
| FR14 | **Tool Accessibility** | 80% | Being supported constantly |
| FR15 | **Navigation** | 100% | Shows also where is the user, in terms of test location |
| FR16 | **Dynamic Content** | 0% | Left for future development, outside of the project scope |
| FR17 | **Test Criteria** | 100% | Adjusted by the client |
| NF1 | **Short Criteria.** Was replaced later on with **FR14** | 100% | Was replaced by **FR17** |
| NF2 | **Gamification** | 0% | Left for future development, outside of the project scope |
| NF3 | **Colour Scheme** | 100% | Supported everywhere |
| NF4 | **Tool Accessibility**. Was replaced later on with **FR14** | 80% | Object highlight doesn’t work |
| NF5 | **Speed** | 100% | AngularJS allows dynamic content change |
| NF6 | **Adaptability** | 100% | All tests are independent from each other and placed in blocks |
| NF7 | **Sequence of Tests** | 100% | Followed as it is in the Web2Access website |

## Appendix I: Adjusted Criteria Text by the Client

***Note:*** Shortened version of criteria written by the client are highlighted in **Grey:**

**Test 1: Login, Sign Up and Other Forms Accessible**

1. Failure with screen reader and keyboard - Lacks labels to forms.
   1. **Fails all AT access, feedback & timing**
2. Failure with screen reader (e.g. CAPTCHA without alternative or inaccessible forms).
   1. **Fails TTS access, has CAPTCHA alternative**
3. CAPTCHA alternative offered or some accessible forms but some labels may be misleading.
   1. **Lacks feedback, but has AT access**
4. Simple, accessible forms with clear labels e.g. 'username (email)' and 'password'.
   1. **Easy access, good feedback**

**Test 2: Image ALT tag**

1. None, detrimental to understanding of content. No option to add alt-tag if uploading image to web pages.
   1. **No alt tags or explanations**
2. Inadequate/sparse alternative text even to actual website images not just those added by users.
   1. **Inappropriate/confusing alt tags**
3. Alternative text offered but lacks brevity or clarity e.g. image of duck.
4. **Adequate alt tags but can lack clarity**
5. Acceptable alternative text throughout
6. **Good alt tags and explanations**

**Test 3: Links**

1. Non-defined links such as 'click here' or just 'download'.
   1. **Meaningless, duplicate & broken links**
2. Non-defined links such as 'click here' or just 'download', but with explanatory title attributes.
   1. **Non-defined ‘click here’, duplicates**
3. Most links understandable or provided in sentences. May have some duplicates.
   1. **No broken or duplicates, some unclear links**
4. Links fully appropriate, used throughout the site plus alternative navigation element.
   1. **Skip nav and clear links throughout**

**Test 4: CSS Removal**

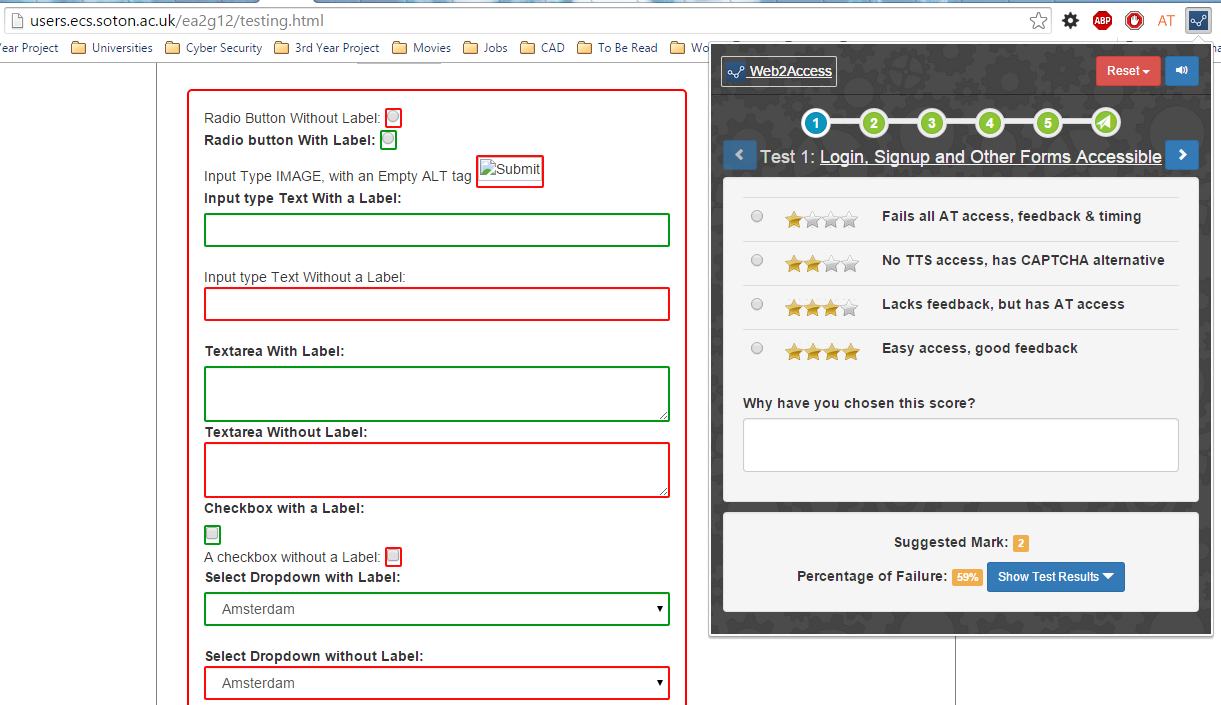
1. Page is unusable.
2. **Page is unusable**
3. Content accessible.
4. **Content available but unordered**
5. Content and navigation accessible.
6. **Content ordered / navigation difficult**
7. Fully accessible with correct document structure.
8. **Content and structure retained**

**Test 5: Appropriate Use of Tables**

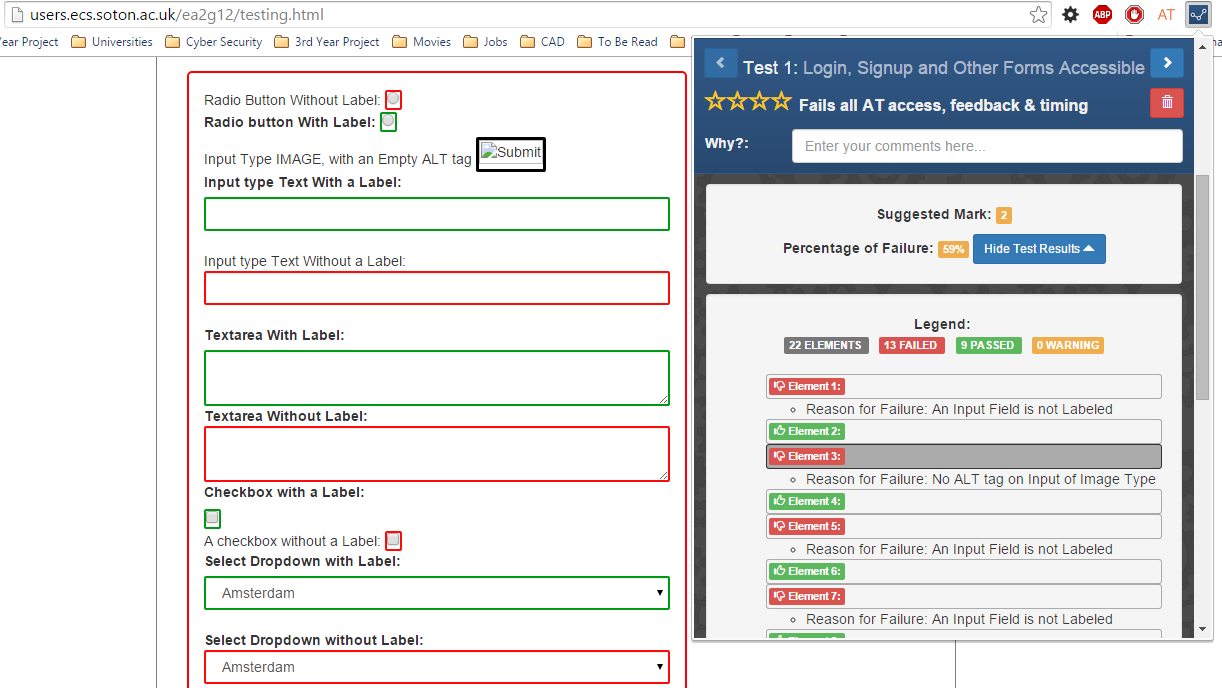
1. Page layout is built using tables and access is poor
2. **Layout using tables / access poor**
3. Data tables, if used, have no headings. Layout tables do not impact on screen reader.
4. **Data tables lack row/col headers**
5. Data tables incorrect layout. Navigation with a screen reader possible with effort.
6. **Some data not associated with headers**
7. Page layout does not use tables and/or headed tables are used to present data.
8. **Tables have headers & associated data**

## Appendix J: The Final Screenshots of the Google Extension

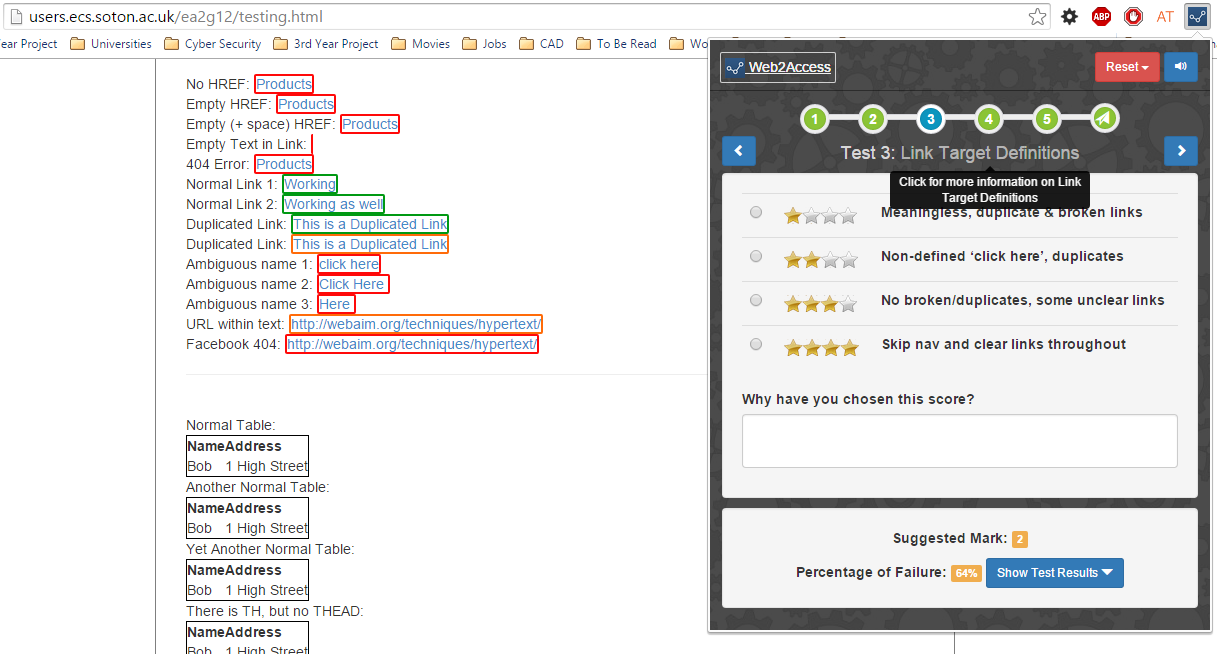
* Testing performed on the following page: [www.users.ecs.soton.ac.uk/ea2g12/testing.html](http://www.users.ecs.soton.ac.uk/ea2g12/testing.html)



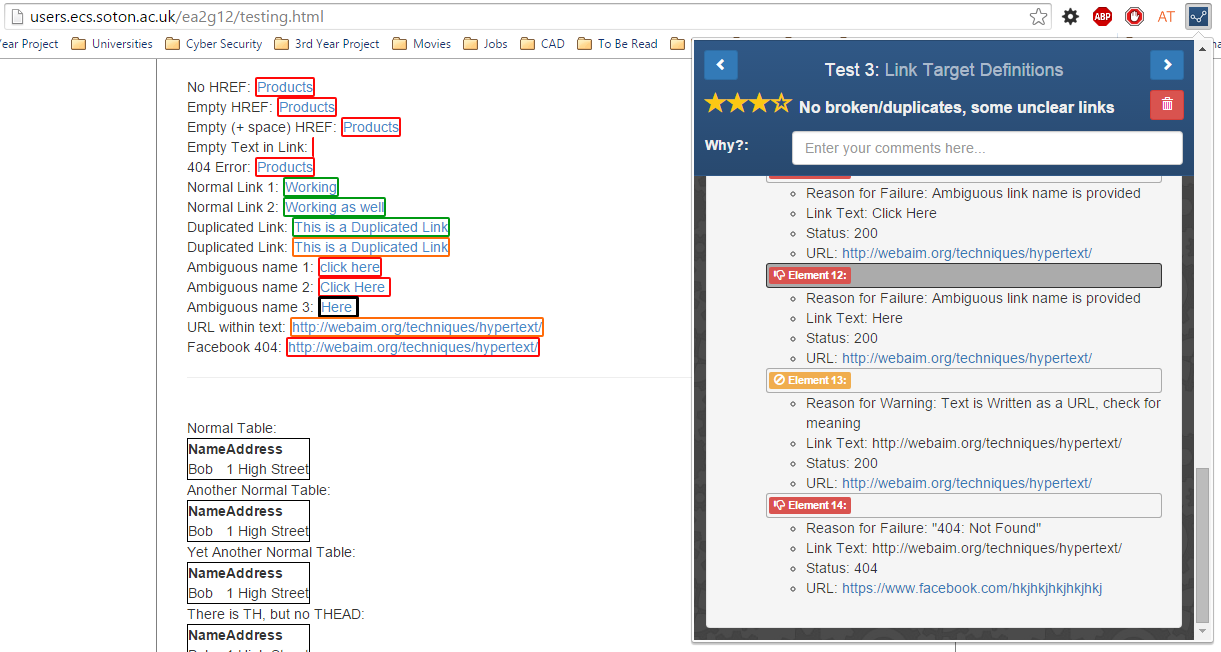
**Test 1: Login, Signup and Other Forms Accessible (a)**



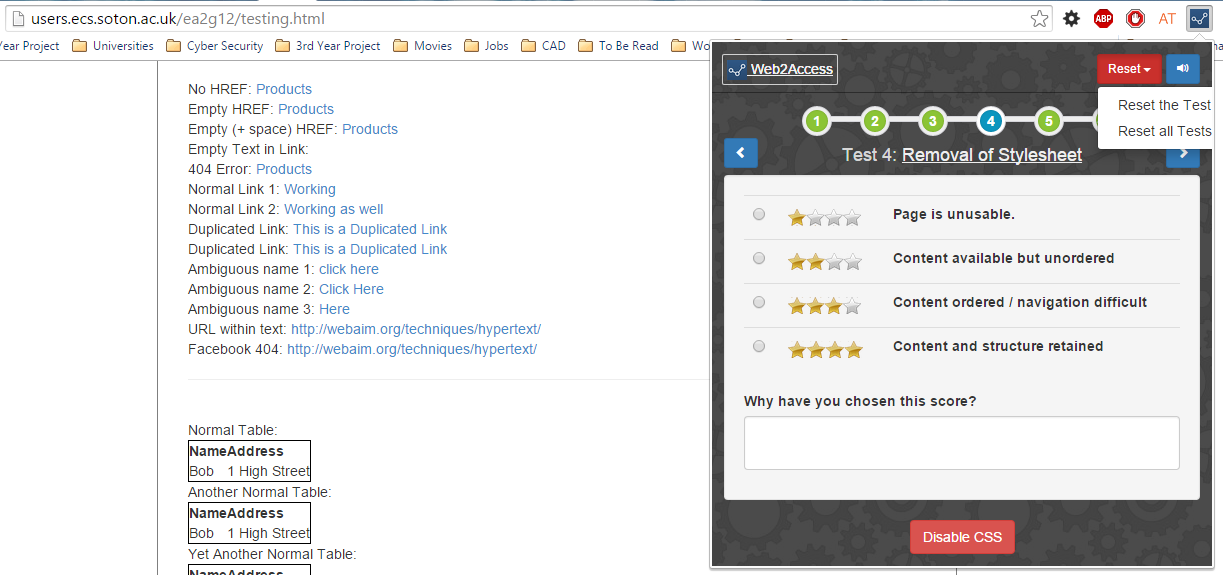
**Test 1: Login, Signup and Other Forms Accessible (b)**



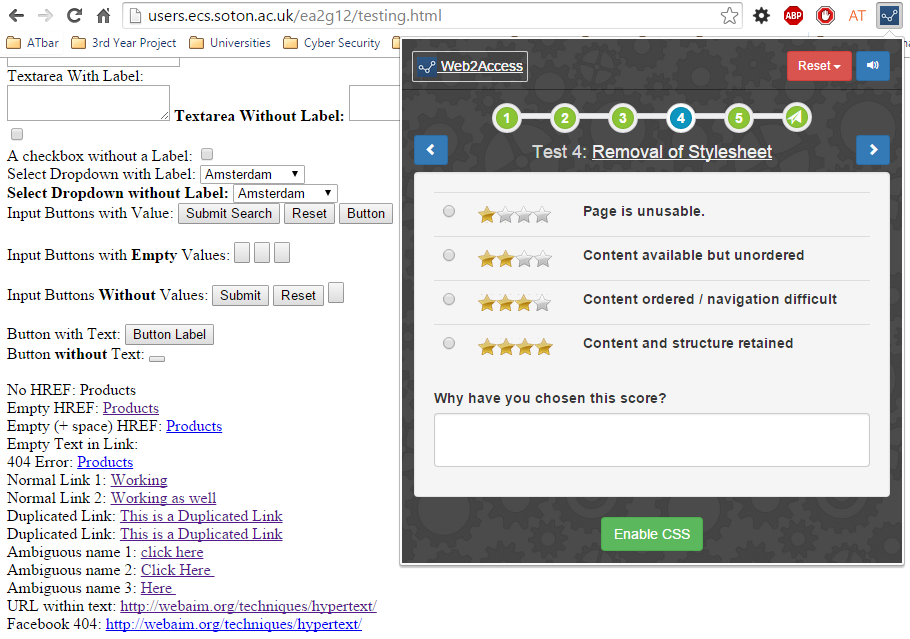
**Test 3: Link Target Definitions (a)**



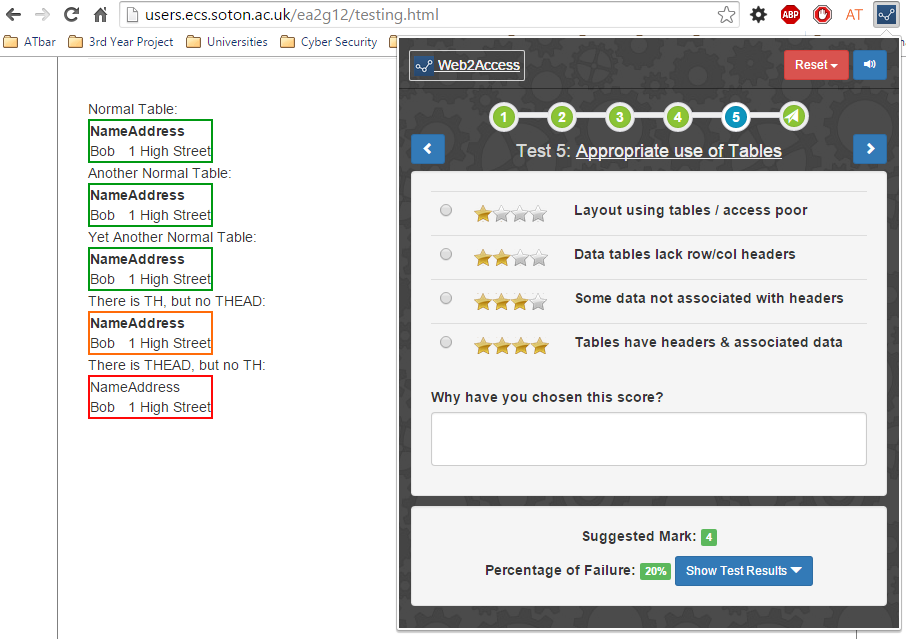
**Test 3: Link Target Definitions (b)**



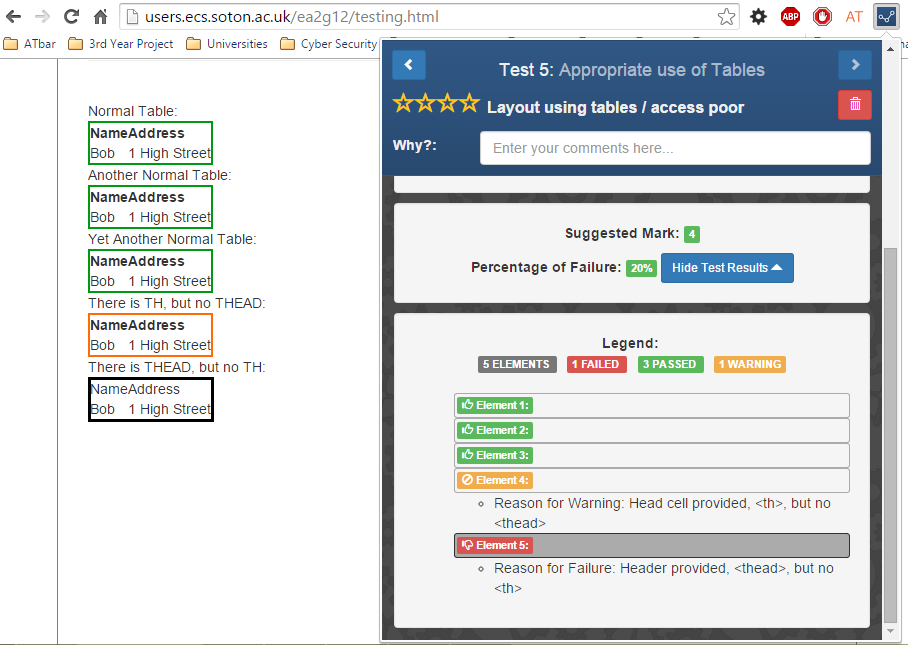
**Test 4: Removal of Stylesheet (a)**



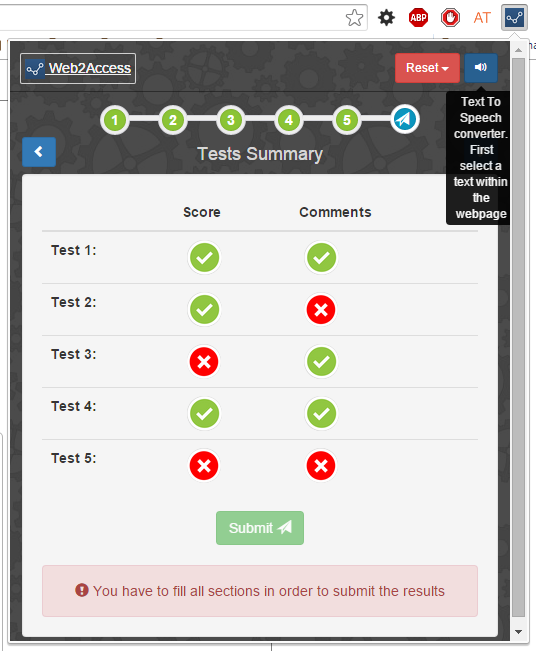
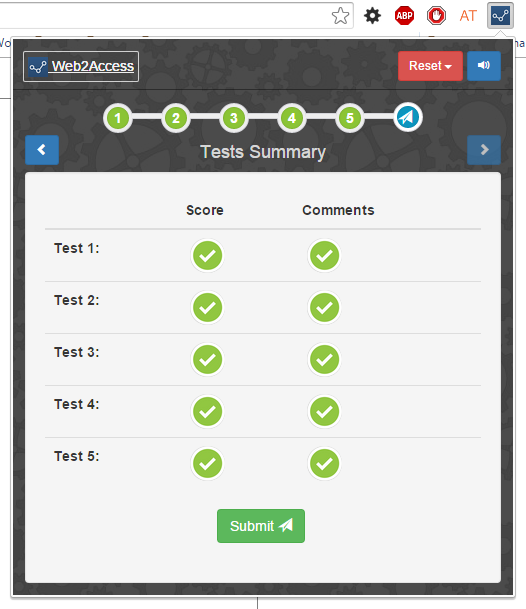
**Test 4: Removal of Stylesheet (b)**



**Test 5: Appropriate use of Tables (1)**



**Test 5: Appropriate use of Tables (2)**

**Tests Results Summary**

## Appendix K: Google Extension Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Accessibility Test | No. | Description | Test Result |
| Test 1: Forms | | | |
|  | **1.** | Radio button with/without label | Passed |
|  | **2.** | Input type image, with/without empty alt tag | Passed |
|  | **3.** | Input type text, with/without a label | Passed |
|  | **4.** | Textarea with/without a label | Passed |
|  | **5.** | Checkbox with/without a label | Passed |
|  | **6.** | Select dropdown with/without a label | Passed |
|  | **7.** | Input type buttons, with/without a value | Passed |
|  | **8.** | Buttons, with/without text | Passed |
| Test 2: Images | | | |
|  | **1.** | Image without alt tag | Passed |
|  | **2.** | Image with empty alt tag | Passed |
|  | **3.** | Image with empty (+ space) alt tag | Passed |
|  | **4.** | Image with filled alt tag | Passed |
| Test 3: Links | | | |
|  | **1.** | Link without a href attribute | Passed |
|  | **2.** | Link with empty href attribute | Passed |
|  | **3.** | Link with empty ( + spaces) attribute | Passed |
|  | **4.** | Link with empty text | Passed |
|  | **5.** | Link to a 404 page | Passed |
|  | **6.** | Normal link | Passed |
|  | **7.** | Duplicated link | Passed |
|  | **8.** | Ambiguous link name (‘click here’) – case insensitive | Passed |
|  | **9.** | Ambiguous link name (‘click here’) – case sensitive | Passed |
|  | **10.** | Link with a URL as a text | Passed |
|  | | | |
| Test 4: CSS | ***Note*:** This is a manual test that cannot be automated. | | |
|  | **1.** | Enable/disable CSS button toggle | Passed |
|  |  |  |  |
| Test 5: Tables | | | |
|  | **1.** | A table with <thead> and <th> | Passed |
|  | **2.** | A table with <thead> but without <th> | Passed |
|  | **3.** | A table with <th> but without <thead> | Passed |
|  | **4.** | A table without <thead> or <th> | Passed |

## Appendix L: Evaluation Tasks

1. **Start**
   * Start the Web2Access Google Extension
2. **Navigate**
   * Navigate through all Tests available within the tool using arrow buttons and familiarize yourself with the system
3. **Test Summary / Details**
   * Find the automated results for a specific test
4. **Marking**
   * Based on the results found, mark the website (for that specific test) on the 4-scaled marking system and justify your choice in the comments box.
5. **Hide/Show**
   * Hide the detailed test results in order to minimize the size of the Google Extension
6. **Searching**
   * Find on the website the specific form element/image/link/table marked FAILED in the extension results.
     + **Hint**: hover over the failed element in the extension
7. **Filtering**
   * Filter Test results/details by different category, e.g. Failed, Warning and Success
8. **Reset/Delete**
   * Reset/Delete results for a specific test or all tests simultaneously.
9. **Tests Summary**
   * View the summary of the tests completeness and using it, navigate to those that you didn’t finish
10. **Submit**
    * Submit your test results to the Web2Access service

## Appendix M: User Interface Problems mentioned during the Evaluations

|  |  |  |
| --- | --- | --- |
| Evaluators | No. | Problems |
| Evaluator A | | |
|  | **1.** | Element highlight is not intuitive, put an con to trigger hover |
|  | **2.** | Put filtering instead of sorting the errors |
|  | **3.** | Put Failed elements first, don’t waste for none |
|  | **4.** | Loading bar in the Test 3 Links, because it takes too much time |
|  | **5.** | Show multiple errors for the element of there are |
|  | **6.** | Hide suggestion when selected another mark |
| Evaluator B | | |
|  | **1.** | Test 4 needs more description |
|  | **2.** | The Element hover is not intuitive |
|  | **3.** | Provide Filter for the errors, warnings |
|  | **4.** | Short Tutorial for the tool, either on the Web2Access or the Google Web Store |
| Evaluator C | | |
|  | **1.** | Collapse the errors into similar ones |
|  | **2.** | Resetting the tests when you close them is bad, change it |
|  | **3.** | onclick=”scrollTop()” allow at least 100px from the top of the window for the websites with fixed navigation bar |
|  | **4.** | “Next” button at the bottom of the test similar to how it was at the top |
|  | **5.** | Problem with the Test 3 numbering: passed = -17; it is impossible |
| Evaluator D | | |
|  | **1.** | Test 5: Tables, need to check for tables where headers are on the side |
|  | **2.** | Link with explanations at the top of the collapsed errors |
|  | **3.** | Show different types of errors for an element if it has failed in more than one category |
|  | **4.** | Bad error description in Test 5: Tables. Change it. |
| Evaluator E | | |
|  | **1.** | Provide positive feedback, so the user knows that the system worked. Ex: “your test results has been saved” |
|  | **2.** | Hard to choose the radio button, put onchange event on the whole criteria so when you click on the text the radio button is selected |
|  | **3.** | Test 2: Images; check for images whose ALT tag starts with “Picture of…”, “Image of…”, “Figure of…”, etc. and put a Warning highlight on it. |

## Appendix N: Participant Information Sheet and Consent Information

Participant Information Sheet

|  |  |  |
| --- | --- | --- |
| Ethics reference number: **ERGO//14256** | Version: 1.3 | Date: 2015-03-15 |
| Study Title: Automated Web Accessibility Checking Tool for Web2Access | | |
| Investigator: Eldar Alasgarov | | |

**Please read this information carefully before deciding to take part in this research. If you are happy to participate you will be asked to give your verbal consent.**

**What is the research about?**

This study is for a third year student project for the award of BSc Information Technology in Organisations. The researcher named above is conducting research in order to evaluate the prototype of the system, its usability and user experience. The purpose of the study is to design and build a software solution to aid stakeholders perform faster web accessibility checks and contribute to the ongoing research process. The questions asked are mostly related to the user interface of the system and its functionality.

**Why have I been chosen?**

You have been chosen because you have been identified as being a potential stakeholder in Web2Access project.

**What will happen to me if I take part?**

You will be asked to perform several basic evaluation tasks on the prototype of the system and answer predefined questions regarding the tasks performed in a structured interview, which will be followed by a basic discussion with open questions to gather qualitative data. The study will take no longer than 30 minutes

**Are there any benefits in my taking part?**

Successful implementation of the Automated Accessibility Tool for Web2Access may benefit all stakeholders, i.e. academics, developers and people interested in the Web Accessibility.

**Are there any risks involved?**

There is no physical risk or deception involved in this study.

**Will my participation be confidential?**

All data will be treated anonymously and in compliance with the Data Protection Act and the University Policy. Data will be stored in a password protected computer that only the researcher will have access to. All data will be destroyed by the end of the project, 28th of April.

**What happens if I change my mind?**

You have the right to withdraw from the evaluation process at any time and request that any response or data collected from you to be destroyed without your legal rights being affected.

**What happens if something goes wrong?**

Should you have any concern or complaint, please contact the FPSE Office ([fpse-student@soton.ac.uk](mailto:fpse-student@soton.ac.uk)) or any authoritative body such as Research Governance Manager (02380 595058, [rgoinfo@soton.ac.uk](mailto:rgoinfo@soton.ac.uk))

**Where can I get more information?**

Please contact Eldar Alasgarov (07413187333 or [ea2g12@soton.ac.uk](mailto:ea2g12@soton.ac.uk))

Consent Information

|  |  |  |
| --- | --- | --- |
| Ethics reference number: **ERGO//14256** | Version: 1.3 | Date: 2015-03-15 |
| Study Title: Automated Web Accessibility Checking Tool for Web2Access | | |
| Investigator: Eldar Alasgarov | | |

*Participants are asked to indicate their agreement to the following statements.*

I have read and understood the Participant Information and have had the opportunity to ask questions about the study.

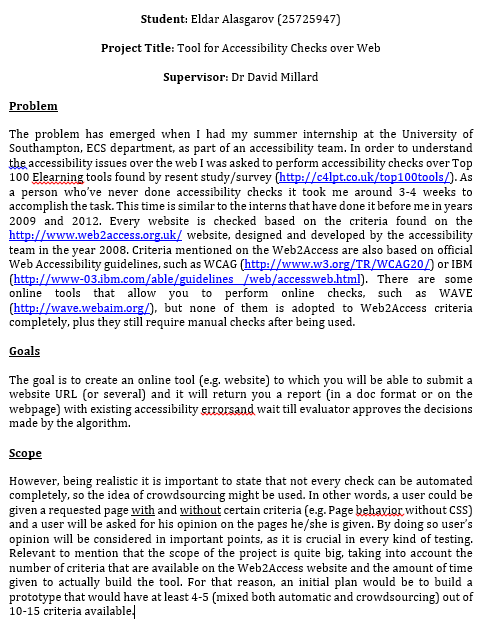
I agree to take part in this study.

I understand my participation is voluntary and I may withdraw at any time and for any reason.

## Appendix O: Structured Interview Questions

1. **Minimize the effort required to perform the test.**
   1. Was the effort required to do the tests minimized? If yes how, if no how it could be minimized better?
   2. What were the main obstacles in performing the tests previously? Multiple screens, multiple windows, multiple tools?
   3. Any other suggestions on how the effort could be minimised?
2. **For the experienced, speed up the process; for the inexperienced, make it actually feasible.**
   1. How long did it take you to finish all the tests with old methods?
   2. How long it took you to finish the tests with the new method (Google Extension)?
   3. Was the tool fast in response, if ‘No’, was it a specific part within the tool or overall performance?
   4. Imagine if you are a novice/non-expert user, would you be able to work your head around this tool and understand it, for example, error messages for an element.
3. **Ensure tool specificity and completeness.**
   1. How would you describe the completeness of the system, in terms of the number of guidelines being supported for each of the tests?
   2. Some accessibility tools just highlight the problematic code, when others have various tests for several possibilities, hence different error messages for every failed element. The latter was used in the Google Extension.
      1. Which way you think it is better? If latter, Google Extension way, is the error messages provide contextual and helpful suggestions?
   3. How accurate were the test results from the Google Extension in comparison to the ones from old methods?
   4. How many of the problems reported by the tool are the actual problems? Please estimate, no exact numbers are required.
   5. Do you agree with the problems reported, if not, why?
4. **Secure more participants with Web2Access.**
   1. Do you think this tool will help to get more participants to the Web2Access project? If not, could you suggest any other solution?
5. **Balance between automatic and manual work.**
   1. Did the suggested marks for the semi-automated tests confuse, change your opinion or effect your mark choice in any manner, if ‘Yes’, how big was the effect?

## Appendix P: Original Project Brief



## Appendix Q: Table of Contents for the Design Archive File

* **css**
  + **app**
    - content.css
    - popup.css
  + **lib**
    - *Mainly Bootstrap related files*
* **img**
  + *Background images and icons*
* **js**
  + **app**
    - app.js
    - background.js
    - content.js
    - popup.js
  + **lib**
    - *AngularJS, Bootstrap, jQuery, and other libraries and plug-ins*
* **tests**
  + test1.html
  + test2.html
  + test3.html
  + test4.html
  + test5.html
  + test6.html
* manifest.json
* options.html
* popup.html

1. <http://www.web2access.org.uk/> [↑](#footnote-ref-1)
2. <https://eternalsunshineoftheismind.wordpress.com/2013/03/page/41/> [↑](#footnote-ref-2)
3. <https://developer.chrome.com/extensions> [↑](#footnote-ref-3)
4. Application Programming Interface – a set of protocols, functions and tools for building applications [↑](#footnote-ref-4)
5. <http://www.sublimetext.com/3> [↑](#footnote-ref-5)
6. <https://github.com/Eddikos/3rd-Year-Project> [↑](#footnote-ref-6)
7. <http://webaim.org/articles/> [↑](#footnote-ref-7)
8. <https://developer.chrome.com/apps/tts> [↑](#footnote-ref-8)
9. <https://api.jquery.com/scrollTop/> [↑](#footnote-ref-9)
10. <http://www.w3schools.com/jsref/met_win_print.asp> [↑](#footnote-ref-10)
11. <http://users.ecs.soton.ac.uk/ea2g12/testing.html> [↑](#footnote-ref-11)
12. <http://wave.webaim.org/> [↑](#footnote-ref-12)
13. <https://api.jquery.com/scrollTop/> [↑](#footnote-ref-13)
14. <https://developer.chrome.com/extensions/storage> [↑](#footnote-ref-14)