

# IAPT Group 12 Report

David Morris

Killian Murphy

Tadas Paliulis

Zain Razi

April 30, 2014

## Contents

<b>1 Reliability and Maintainability</b>	<b>1</b>
<b>2 Design Rationale</b>	<b>4</b>
<b>3 Collaborative Heuristic Evaluation</b>	<b>8</b>
<b>4 Task-Based User Evaluation</b>	<b>11</b>
<b>5 Accessibility Evaluation</b>	<b>15</b>
<b>Appendices</b>	<b>19</b>
<b>Appendix A Figures</b>	<b>19</b>

## 1 Reliability and Maintainability

Our approach to reliability and maintainability took its cue from the purpose of the system.

The purpose of the system:

- On one level, to explore a design for an adaptable cookery book.
- On another level, to serve as a focus for building and demonstrating an understanding of techniques used for evaluating and improving interactive systems.

These purposes place the engineering of the system as a peripheral concern: it is important that the system be sufficiently well-built that it is easy to modify and doesn't fall over while being evaluated, but making an application which is bulletproof in the face of malicious user input and whose code crackles with craftsmanship would be misspent effort.

Given this, we have generally followed lightweight processes intended to produce a reasonably robust and maintainable system without impeding the flexibility required to address the interaction design and usability focus of the assessment.

### 1.1 Coding Standards

When writing the code, we followed a number of principles to aid in ensuring reliability and maintainability.

**Naming** We have striven to follow a consistent naming scheme for classes, functions, and variables. This reduces mental overhead and confusion when working with the code, improving maintainability.

**Consistent formatting** Similarly, we have followed a consistent formatting convention (enforced by tools) when working on the codebase. As with naming, this reduces mental overhead and confusion when working with the code, improving maintainability.

**Commentary** All of the files, classes, and functions we have written have PHPDoc comments explaining them. This documentation is critical for maintainability; it also helps reliability, as it requires a clear understanding of the purpose of the code, which demands simple, clearly-written code.

**Fragile PHP features** We were careful to avoid some of the easy-to-misuse features of PHP, such as the `__get` and `__set` magic methods, references (because of the weird way they taint the symbol table), `$$`, etc.

Doing this has improved the reliability and maintainability of our system by avoiding the surprising and error-prone ‘mysterious action at a distance’ they cause.

## 1.2 CSS Preprocessing

CSS is missing a number of abstraction features which are important in keeping stylesheets maintainable, such as the ability to define variables (for colours, sizes, etc), and functional units (such as margin-based block centring). To improve the maintainability of our system, we used the Sass CSS preprocessor[1] to generate pure CSS from our more-maintainable SCSS stylesheets.

## 1.3 Version Control

### Choice of `git` over Subversion

Although a Subversion repository was provided for our use by the department, we chose to use a more modern version control tool, `git`. The principal reasons for this were as follows:

- Data integrity: with inexperienced users, subversion can be fragile and occasionally trashes its working-copy metadata.
- Team fit: our team is inherently distributed and mostly works separately, a natural fit for `git`.
- Safe merging: the `git` model of merging two commits is less fragile than the subversion model of merging the remote onto an uncommitted working copy, which can cause data loss.

### Version control process

Given the small size of the project, we opted for a lightweight version control process. Most of the time, we followed a centralised workflow, as with subversion, as this was sufficient and appropriate for the experience level of the team. For similar reasons, we mostly eschewed ‘advanced’ features such as feature branching, history rewriting, etc.

In specific instances, for more complex and sweeping changes, we used a slightly more rigorous process, working in dedicated branches which were subsequently reviewed and merged using GitHub’s pull request system[2].

In practice, we were satisfied most of the time with a simple no-branching centralised model. It was very useful, however, to have seamless access to the more powerful tools afforded by `git` in specific situations.

Our version control process, although simple, establishes a clear audit trail for the codebase; this greatly improves the maintainability of the system by allowing future developers to understand the history of, and rationale for, each piece of functionality, and provides a valuable source of additional documentation.

## 1.4 Issue Tracking

### Choice of Issue Tracking tool

We used GitHub’s issue tracking system. GitHub was already a natural place to host the principal copy of our repository, and their issue tracking tool is simple but reasonably full-featured, and has good repository integration.

## Issue Tracking process

We made extensive use of our issue tracking tool to keep track of bugs and outstanding work.

At specific points in the development cycle—particularly prior to each stage of evaluation—we used milestones to collate issues and ensure that we went into each evaluation with the system in a known, well-defined state.

Our issue tracking process has contributed significantly to the reliability of the system by ensuring that no work ‘fell through the cracks’, and that no bugs, once identified, went unfixed.

If we (or someone else with access to our issue tracking system) were to continue developing the system, our issue tracking process would also contribute significantly to maintainability, providing documentation and a rationale for much of the work done, and a powerful insight into the system.

## 1.5 Sample data

To allow flexibility in the data model and database schema used, we wrote some small scripts that would take naturalistic YAML descriptions of recipes (basically a one-to-one translation of the provided sample recipes) and populate the database from them. This made it easy to drop and recreate the database as we experimented with and refactored the data model.

This improves reliability by guaranteeing that the database will contain no invalid data; it also improves maintainability, by making it easy to refactor the database or extend that data model.

## 1.6 Testing

Most of the testing we did was effectively informal acceptance testing (with a side of integration testing); because of the amount of time we spent evaluating the website from a usability and interaction-design perspective, we effectively and thoroughly exercised every part of the application on a regular basis. While this testing was not specifically directed towards ensuring reliability, the effect was to demand a high level of reliability, and highlight any robustness or reliability issues immediately.

However, we also did some more explicit testing to help with maintainability.

### Integration Testing

As noted above, most of the testing was acceptance testing of some form. However, we found a place for some lightweight integration testing to catch casual errors.

Our integration tests take the form of some very minimal smoke tests that check that the various URLs associated with our application are *a)* reachable and *b)* do not contain any obvious errors emitted by the PHP interpreter.

These goals are met by `app/test/smoke.sh`, which uses basic unix utilities to check these properties for the various important URLs in the application.

Having these tests has helped to improve the reliability of the system by making it easy to ensure a basic level of functionality across the application. They also help with maintainability by allowing more confident refactoring, acting as a safety net to catch heinous errors early.

### Unit Testing

Some form of unit-level testing is a fundamental part of any engineering process. However, the complexity of CodeIgniter initialisation and the tight coupling of our application to the database make it awkward to use classic xUnit-style strongly-decoupled tests.

Our compromise was a page within the CodeIgniter site (`index.php/test`) which, when visited, executes a number of simple tests which exercise the model layer. Given the database coupling, these really sit somewhere between unit and integration tests, but we felt this was acceptable given the lightweight nature of our model layer. Rather than an in-depth and exact verification of precise behaviour, these tests also function more as smoke tests, making it easy to exercise the whole of the model layer at once to catch a broader range of model-layer issues than the integration tests above.

These tests have increased reliability by ensuring that the model layer works, and increase maintainability by allowing us to make changes with confidence.

## 2 Design Rationale

### 2.1 System Overview

The final recipe-presentation web application was the product of a many-step process. These steps, in order, were:

- Research into similar existing services, carried out to provide insight into the downfalls and successes of applications produced by others;
- Analysis of the application specification;
- Devising of user personas and scenarios for those personas, created to offer something of a user-centred perspective to the design process;
- Formation of formal requirements for the application, these providing an easily-verified set of criteria laying out the minimum functionality required for the application;
- Creation of an interactive first prototype;
- Collaborative heuristic evaluation of the first prototype;
- Redesign of the prototype to rectify problems uncovered in the CHE, and to bring the prototype more in line with both functional requirements and the goals implicit in the user scenarios;
- Task-based user evaluation of redesigned prototype;
- Final adjustments of the prototype to alleviate issues raised in the TBUE, and to ensure that goals and functional requirements were met;

From the initial research into other services and analysis of the specification, many personas and scenarios were created. The purpose of these was to both identify the potential users of the system to be designed and to aid in the formation of formal functional requirements. The personas and scenarios follow.

#### Jennifer

- Persona** Female, early-twenties, novice cook, computer-literate.  
**Scenario** Starts cooking, becomes distracted. Comes back to checked off recipe steps.

#### Dale

- Persona** Male, middle-aged, accomplished cook, computer literate.  
**Scenario** Hosting dinner party, needs vegetarian dish, preparation needed to time guests' arrival.

#### Bob

- Persona** Male, teenager, inexperienced but keen cook, computer literate.  
**Scenario** Begins viewing a recipe in narrative, finds it too difficult, changes to segmented.

#### Mike

- Persona** Male, middle-aged, no cooking experience, computer illiterate.  
**Scenario** Shopping for ingredients, forgot to make a list, needs a list of ingredients for recipe.

#### Mavis

- Persona** Female, elderly, experienced cook, computer illiterate.  
**Scenario** Wants to use her recently purchased tablet to look up recipes.

#### Andy

- Persona** Male, early-twenties, intermediate cook, computer literate.  
**Scenario** Begins using narrative view, switches to step-by-step for more information, finds and checks off the step he was at.

Following the personas and scenarios came the functional requirements. These were laid out to provide clear-cut goals which needed to be achieved by the application in order for it to provide a satisfactory quality of service. These requirements were split into two headings, each heading reflecting an essential component in the interactive system:

**Recipe Browsing** Users must be able to:

- search for recipes using a full or partial recipe name;
- view a list of all recipes that are present in the application's database of recipes;
- filter any list of recipes using specific filtration criteria.

**Recipe Viewing** Users must be able to:

- see general information about a recipe both when browsing multiple recipes and when viewing a single recipe;
- see the ingredients required to complete a recipe;
- see the instructions that need to be followed in order to complete a recipe;
- change the level of detail in which recipe instructions are presented to them;
- set the default level of detail in which all recipe instructions are presented to them;
- mark instructions as completed through some interaction.

Although all of the functional requirements were met in the final version of the application, the functionality of the system that has been produced is a little richer than the functionality suggested by the functional requirements. The complete functionality of the system consists of that contained in the functional requirements plus the addition of an interactive help system. Users have the option, at any time, of clicking a 'help' element. This will provide them with help messages - these messages describe the key features of any page on which the help is requested, and can be dismissed by the user with a simple click.

To aid the making of design decisions, our group attempted to distill and combine Shneiderman's Principles of Interaction Design[3], Don Norman's Design Principles[4] and Tog's First Principles of Interaction Design[5] in order to make them easier to work with. These criteria have been identified as a solid starting point for any user interface.

## Design Criteria

1. Consistency;
2. Informative feedback to the user;
3. Straightforward reversal of actions;
4. Promoting internal locus of control;
5. Low short-term memory load;
6. High feature visibility;
7. Information readability;

Two pieces of complex functionality were chosen based on the magnitude of their contribution to the application's functionality as a whole. They each fall under a distinct heading in the functional requirements, and thus had to fulfil different sets of functional requirements. The functionality chosen was 'viewing a recipe' and 'filtering lists of recipes' - full design choice justification for each follows.

## 2.2 Viewing a Recipe

The following is the design rationale behind the first piece of complex functionality found in the produced application: the process of viewing a recipe.

When designing the recipe view we tried to maintain a strong sense of consistency to the page, as identified in criteria 1, all fonts are uniform within their respective page groups (see figure 4). We decided against bolding the names of ingredients in the recipe instructions even though we do bold the ingredient amounts. We felt this would just introduce unnecessary clutter breaking the sense of consistency and perhaps provide a false feedforward suggesting to the user that the name is pressable when it is not.

Similarly we maintained information readability, criteria 7, by using consistent and readable font sizes and spacing. Our primary rationale behind this coming from the personas and scenarios, in particular Mavis. Not only is readability an issue for elderly (due to potentially reduced eye sight) but is also a major issue for users on mobile devices. As her scenario outlines her recent purchase of a tablet, we decided to pay extra attention to readability to ensure our website be usable on mobile without us having to create a separate web client.

The only exception is the difference in spacing between ingredients and instructions. As discussed in Jennifers scenarios (and others) we envisage users wanting to use the site before they are ready to cook. We believe that they will be looking up ingredients for dishes while shopping. It is for this reason we purposely break down the readability between the two to cement them as separate elements to the page making it easier to quickly look up the ingredients.

Short-term memory load, criteria 5, was identified as a key issue early on, as outlined in Mike's persona part of our user base are likely to be very novice cooks. They may well be using the website whilst cooking and will need information delivered to them in digestible chunks so as to not overwhelm them. We tried to ensure this by providing adequate spacing between instructions especially within the step by step view (see figure 5). The separation encourages the user to read a step and then process the information before moving on to the next. We feel the spacing is not enough to make the user think it is not belonging to the same group and at the same time is sufficient enough to ensure low short-term memory load. The user may well be dipping in and out of reading the page and thus giving them information in small chunks will be beneficial, this is shown through Andy's scenario where he refers to the step by step view while cooking. The separation means he can easily scan the separate steps to find the one he should be on.

On way in which criteria 6, high feature visibility, was upheld was by ensuring the buttons to change the type of view are coloured differently to their surrounding elements (see figure 5). Though their relative size is small we feel the colour difference is sufficient enough to draw attention to it without making it the main feature of the page.

Regular users of our website, users potentially like Bob, are likely to be accessing recipes often. We want to provide them with a sense of customisability and control over their experience. By allowing them to choose which type of recipe view the website defaults to we hope to promote internal locus of control. To this effect the view buttons also allow the user to set a default view (see figure 5) and thereby fulfilling principle 4.

Criteria 3 provided an interesting dilemma for our design. The type buttons do not immediately provide straightforward reversal of action for the user, pressing a view button twice will not undo its action. We did this on purpose to stay inline with what our research of other websites has shown us, we think the user will not expect this behaviour and as the other buttons are in close proximity they will use those to undo the action. Though our more tech illiterate users may struggle with this, Mavis not being familiar with established website design patterns as outlined in her persona, we believe the majority of our user base will appreciate it being done this way. As Mavis becomes more familiar with internet usage this will not be a problem for her either, and in fact we should follow the norms of other segmented control patterns so as not to confuse her when she goes to other website.

A more concrete way in which criteria 3 is followed was in the design of the instructions ticking off feature. We do provide an easy reverse for ticking instructions, as users would expect pressing the element again reverses the action. The ticking off feature has also been designed to have high feature visibility. Empty tick boxes are always present which already suggest to the user that they can be ticked off, but further to this when a step is moused over it is highlighted providing the extra feedforward to further clarify its functionality.

## 2.3 Filtering Lists of Recipes

The following is the design rationale behind the second piece of complex functionality found in the produced application: the process of filtering lists of recipes.

After examining the personas and scenarios created by our group we came to the conclusion that easy recipe browsing and access is a recurring theme and decided that it is a high priority issue. Most cooking websites have a high volume of recipes, therefore robust tools are needed. Our solution to this problem was the introduction of the filtering menu. The multiple available choices enable the site visitors to get especially specific with their recipe needs. It shows up whenever user is browsing the recipes or after performing a search.

While designing the filters we used Shneidermans Eight Golden Rules of Interface Design and Don Normans design principles.

Based on our research of other recipe websites and phone applications available on the web as well as our scenarios we distilled the following available filter types and their selections were chosen as follows. First filter is Dish type, with available selection of Main, Salad, Side and Dessert. In our scenarios, personas are cooking for very different occasions. The loner Mike might be trying to get a simple main course dish done for himself, while Dale might desire to prepare a dessert for his friends. Next is Dietary restriction, which allows choosing between Vegetarian, Vegan, Kosher and Halal. We considered this especially important, since it caters to users with extremely varying tastes. For example, in one of our scenarios Dale is preparing for a dinner party. One of his friends is a vegetarian, therefore Dale wants to make sure that at least one of the dishes he prepares is vegetarian, so that all of his guests are satisfied. Difficulty filter has the expected choices of Beginner, Intermediate and Advanced, making it easy for the site visitors to narrow down the selection to their desired skill level for the recipes. These choices stem from our created personas such as Mike, who does not have any cooking experience whatsoever and struggles with even the most basic recipes, and Bob, who is on the opposite end of the cooking spectrum, and prefers to use recipes that would challenge his abilities and help him improve as a cook.

The serving information allows the user to specify the number of portions that can be made with the recipe. This filter can be justified by our scenarios and personas. For example, Mike, cooks only for himself, since he lives alone, he only looks for recipes that serve 1. Another example is Jennifer, in one of the scenarios she is cooking a meal for all her housemates and wants to make sure that there is enough food for everyone. The time field allows to choose the maximum time a recipe would take to prepare. This is useful to one of our personas, Dale, who is cooking for his friends and wants to be done with cooking before they come over. The time and serves fields have placeholder text, further explaining what those fields accomplish in regards to the filtering system. This acts as an affordance and helps the user figure out what inputs are expected from him or her as can be seen in figure 6.

The filter menu and the recipes use the same gradient background so that the user would naturally associate the two. The filters work and look exactly the same on both browse page as well as on search results page (except the filtering is done on the search results, rather than the whole database). This provides expected consistency between the pages, which means that the user has to figure out how to use the filters only once.

The filter menu takes up prominent space in the page to immediately draw attention to it. This is done with the intention of making sure that the user does not miss this important part of the website. This acts in accordance with Don Normans principles of high feature visibility and our criteria 6.

The reset button takes back the filter to its default state as well resetting the the recipe results, providing easy reversible actions as suggested by criteria 3. Both the filter and the reset buttons are made to look obviously clickable, which acts as an affordance as described in Don Normans Design Principles. The filters immediately react to changes by the user, offering immediate feedback on the performed operation and fulfilling criteria 2. This also supports the internal locus of control rule described by Shneiderman and us in criteria 4. After a filtering operation is performed, the filter menu choices previously made are still visible in the menu. This allows for easy adjustments and also reduces the short term memory load (criteria 5) while browsing the recipes, because the user does not have to remember the choices made previously.

If after filtering there are no recipes found a helpful message shows up saying that the search came up with no results as illustrated in figure 7 again providing informative feedback and fulfilling criteria 2. This means that the user understands whats happening and shows that the website is still functioning correctly. This is in accordance with the 3rd rule Offer Informative Feedback from Shneidermans Eight

Golden Rules of Interface Design.

The results themselves are displayed in the middle of the screen and are designed to be the centerpiece of the website. There is clear separation between individual results while still maintaining visual consistency in accordance with criteria 1 and 7. The recipes are displayed as information cards, ideally providing enough information at a glance to make an informed choice. This was inspired from our research done on popular cooking web sites available online. Each recipe is clearly separated from each to avoid confusion, however there is still visual consistency amongst them. Clicking on a recipe card leads to the full view of a recipe. To make this obvious, the recipe cards were designed to look like clickable buttons.

### 3 Collaborative Heuristic Evaluation

Our group decided to follow the CHE methodology outlined by Petrie and Buykx in the paper “Collaborative Heuristic Evaluation: improving the effectiveness of heuristic evaluation” [6]. Coming into the CHE session each group member was asked to generate a list of potential problems to present to the group. These were collated and as a group were shown in sequence. Each group member individually rated the problem using the scale shown in figure 1. Care was taken to not get drawn into a discussion of solutions in this session but to instead focus on identifying the problems. Petrie and Buykx suggest including a domain expert in this session, this was not needed as the development team were completing the evaluation. This did show a major flaw in our methodology in the sense none of the group members are particularly experienced user experience evaluators. Petrie and Buykx elude to the fact that we should use user experience experts guided by one or two domain experts, not the other way around and certainly not without any user experience experts. So while this problem is not really avoidable given the nature of our project it does throw into doubt some of the data we generate within this evaluation. Extra preparation will be needed on the part of our team to get as close to the expertise level of actual user experience experts as is realistically possible.

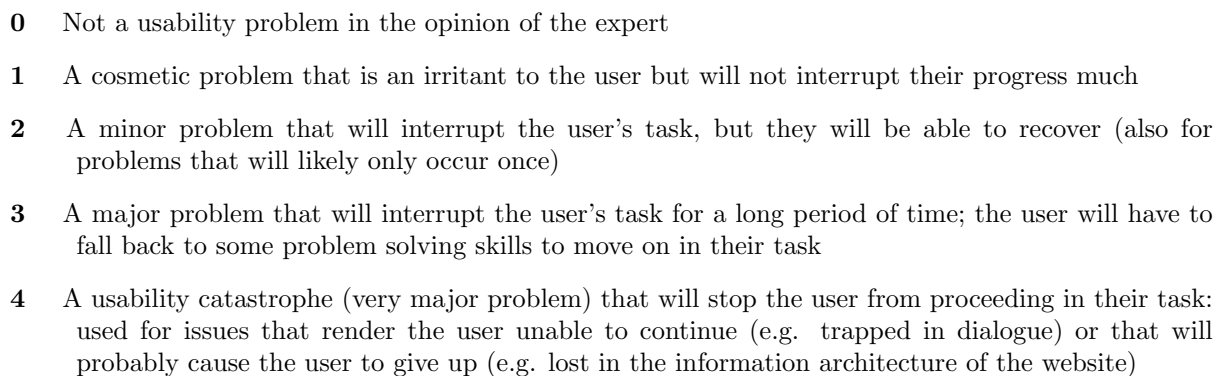
- 
- 0 Not a usability problem in the opinion of the expert
  - 1 A cosmetic problem that is an irritant to the user but will not interrupt their progress much
  - 2 A minor problem that will interrupt the user’s task, but they will be able to recover (also for problems that will likely only occur once)
  - 3 A major problem that will interrupt the user’s task for a long period of time; the user will have to fall back to some problem solving skills to move on in their task
  - 4 A usability catastrophe (very major problem) that will stop the user from proceeding in their task: used for issues that render the user unable to continue (e.g. trapped in dialogue) or that will probably cause the user to give up (e.g. lost in the information architecture of the website)

Figure 1: CHE Evaluation Scale

#### 3.1 Evaluation Heuristics

Our group was given the choice between using Nielsen Heuristics for interactive systems or the Petrie and Power Heuristics for Interactive Web Applications. Even though the Nielsen heuristics are largely more popular we chose to use the Petrie and Power Heuristics. We felt more comfortable using the more fleshed out categories, and what little experience our group had with evaluating heuristics was with the Petrie and Power set. Seeing as the level of evaluation expertise of the group is a major concern it would seem foolish not to go with what we are experienced with and therefore throw away whatever small improvement of the reliability of data the produced by this exercise.



## 3.2 Results

The results for the CHE are contained within figure 2. The task that displayed the majority of severe heuristic issues are ones involving the recipe page, in particular we asked the user to read a recipe and navigate between the different view types. We have decided to highlight the issues that were granted a rating of 2.5 or above, even though all the issues will be tackled these are the highest priority changes and the ones with largest impact.

*Hard to tell where one ingredient begins and the other ends* (see figure 8)

This breaks heuristic 8, provide clear well organised information structures. The information is all present however it is very difficult to see the boundaries between items in the list. The structure does not suggest that the information is a list and as such the users first instinct is to read it as a paragraph, for which it is not grammatically suitable. After reading the paragraph it becomes obvious the information should be presented as a list allowing the user to recover. With an average rating of 2.75 this problem does not stop the user completing their task but in our opinion will definitely slow them down and provide a source of irritation. Coupled with the fact that the recipe list is likely to be referenced often during the process of reading a recipe, this problem is a significant flaw in our prototype.

*No indication that recipe instruction can be checked off* (see figure 9)

Here we are guilty of breaking interactivity heuristics how and why and interactive elements should be clearly distinguished. Though we do provide some feed forward when the cursor is moved over the element we rely completely on the users curiosity, and a certain amount of luck, for the user to discover this functionality. At some point we must tell the user that our site allows them to check off items from the list or provide some visual indicator that the recipe list is not just a list of text but a list of pressable elements. With an average severity rating of 3.5 we clearly see this as a major issue, unless something is done the user is going to completely pass this functionality by.

*No idea which view you are in/press button for view that you are in, nothing happens* (see figure 10)

This does not provide “clear labels and instructions” in relation to both interactive elements and the current presented content and fails to provide feedback on user actions and system progress. The user is not able to definitively know which view they are in, short of figuring it out based on the content which if they are the novice user isn't going to happen. Related to this problem, the fact that nothing happens if you press the button for the view you are currently on is a cause for concern. To the user there is no difference between this button and the others and as such it is reasonable for them to expect something to happen when they press it. When they press it, and nothing happens, there is a complete lack of feedback. The user is not given any indication that their button press has been registered let alone what effect it had on the system.

Both these problems were rated very highly both having an average over 3, though they are serious problems that affect the usability of the site it is possible for the user to use their problem solving skills to complete their task. Thus we decided it did warrant the catastrophic rating even though we will definitely be addressing this issue.

*Changing view: don't know what view change buttons do or what they relate to/don't know what other views are* (see figure 10)

There is a problem with the buttons that change from the different types of recipe view. There is no explanation for what these different views are or the effect they have on the recipe view by switching between them. This breaks the how and why heuristic, obviously the problem lies in not educating the user to what these options are, they could figure out what they do by trial and error and such this problem was rated 3.5/2.5 but again not quite a 4.

*No distinction between instructions* (see figure 9)

Again we are breaking the information architecture heuristic of providing clear, well organised information structures. Very similar to the problem with the ingredients list each step of the recipe is an element in a list but it is presented like one complete body of text. Not a problem in narrative view as

it is mostly one large piece of text but a major issue in step by step and segmented views as the main purpose of the view is to break down a lengthy recipe into steps. The group rated the problem at 2.5, though it slows down the user considerably and is not pleasant to use, the user can still ultimately carry out their goal.

### Look through list of recipes

Problem	Heuristics	Z	T	D	K	Avg
When attempting to find a list of recipies to look at, an option is presented to go to said list but the text does not stand out enough	1	1	1	2	1	1.25
List of recipies not easily readable, not enough separation between items in the list	1	1	1	1	1	1
Within each item there is little distinction between title and body text, not easy to scan for titles.	8	2	2	1	2	1.75
Clicking list navigates user to recipe page, no real indication given to this effect or that the element is clickable.	19	1	3	2	2	2

### Search by Name

Problem	Heuristics	Z	T	D	K	Avg
User not told what to search with ie by name or ingredient	10	2	2	1	2	1.75
No search result: no advice on how to recover	21	3	1	3	2	2.25
With/without search result: search query not shown	16	2	2	1	1	1.5

### View a recipe, check ingredients

Problem	Heuristics	Z	T	D	K	Avg
Ingredients position not quite where it should be, white space on the right	2	1	0	0	0	0.25
Hard to tell where one ingredient begins and other starts	8	3	3	2	3	2.75
No indicator that recipe can be checked off	9, 19	3	3	4	4	3.5
Quantity is not seperated from name	8	2	2	1	2	1.75

### View recipe instructions

Problem	Heuristics	Z	T	D	K	Avg
No idea which view you are in	10	3	3	3	3	3
Numbers not highlighted in narrative view	4	1	0	0	0	0.25

### Changing the view type

Problem	Heuristics	Z	T	D	K	Avg
Don't know what these buttons do, or what they relate to	10	4	4	3	3	3.5
Press button for view where you are nothing happens	13	4	3	3	3	3.25
Click on other views, dont know what other views are	9	2	3	2	3	2.5
Button size too small	1	0	0	0	0	0

### Step by step view

Problem	Heuristics	Z	T	D	K	Avg
No distinction between instructions	8	3	2	2	3	2.5
Not easy remember place in text when looking away	2	2	2	2	3	2.25
Numbers not highlighted in step by step view	4	1	1	1	1	1

### Segmented view

Problem	Heuristics	Z	T	D	K	Avg
No distinction between instructions	8	3	2	2	3	2.5
Not easy remember place in text when looking away	2	2	2	2	3	2.25
Numbers not highlighted in step by step view	4	1	1	1	1	1
No link between ingredients and steps	8	0	1	1	1	0.75

Figure 2: CHE results table.

## 3.3 Application Design Changes

The first part of the redesign addresses the problems with providing clear, well organised information structures, namely the ingredients and recipe instruction lists now have clear spacing between elements within them (see figure 11). Care was taken to ensure the grouping remains intact so that the user can easily tell what is part of the instructions and what is part of the ingredients. To aid this distinction further a different line spacing style was used in both elements, the ingredient list is slightly tighter packed so that it is obvious that text does not continue on from the recipe step to whichever ingredient is level with it.

The larger change is related to the changing view type buttons. The positioning was moved to above the ingredients and recipe instructions. As changing the view changes the way these elements are displayed we thought it best to form a grouping between these and the buttons that control them to suggest to the user that they are related. We are now using a segmented control style of button, the view we are on will now have a visual indicator that it is currently selected (see figure 12). This solves both the problem of the user not knowing what view they are currently on as well as allowing us to disable pressing of the button for the view that is already displayed.

The problem of informing the user what the different view types are and what they do was more difficult to handle. We thought of adding extra labelling to the buttons but felt it would create undue clutter. Instead we opted for a one time tutorial (see figure 13). Either taken upon first visit of the website, where everything will be explained, or if this is declined a shorter version presented when first viewing a recipe. Ultimately we thought this the best way to inform the user of what they different views are and why they might want to use them, the information that needs to be communicated is too complicated to fit on the button structure and only need be shown once and so fits in well with a tutorial system. The user can call up the tutorial again by going through help.

Similarly this tutorial will also be used to tell the user about the sites functionality to check off steps in the recipe. Mousing over a step will cause it to display some feedforward to further cement the idea.

## 4 Task-Based User Evaluation

### 4.1 Methodology

According to Marieke McCloskey in an article she wrote recently called “Turn User Goals into Task Scenarios for Usability Testing” [7], you should use the goals of your system to guide you when creating tasks for task based evaluations. This idea resonated with our group; it follows that the best place to start when designing a good test scenario would be to ensure that it adequately tests that the user can achieve the core goals that they need to. As such we decided to first begin by defining the goals we hope to test.

**Being able to find a recipe** To do this they must be able to browse content of the site in some way and be able to utilise search and filters to find a specific recipe.

**Viewing a recipe** To do this they must be able to effectively manipulate view types and acquire all other information needed to carry out the recipe.

Marieke also defined three key principles to keep in mind to when creating tasks that we endeavoured to follow:

**Make the task relate to real life** She argues that making the task feel real makes the user perform more realistically: if you can suspend the user’s disbelief, you will get more reliable test data. Even if you ask the user to do something they usually wouldn’t, if you can flesh out the scenario enough you can achieve this effect.

**Make them actionable** She describes it as important to make the user actually use the system, it seems obvious but if the task is phrased badly the user could stop trying to actually use the system and instead just describe what they would do to the tester, which doesn’t provide any useful data.

**Avoid clues in task step** (Avoid using terms used on the site). Obviously, if the task contains phrases the user sees on the site they will gravitate towards them, even if that wouldn’t be their normal interaction with the site.

So, ensuring we followed these principles, and that our tasks encompass all the identified goals, the following tasks and scenarios were created:

### Task 1

<b>Scenario</b>	You are an experienced cook, you are looking for a challenge and would therefore only like to cook something that is hard to make.
<b>Instructions</b>	Find the list of available recipes for main dishes that will pose a significant challenge in preparing.

### Task 2

<b>Scenario</b>	You love vanilla slice, but have no idea how to make it.
<b>Instructions</b>	Find a Vanilla Slice recipe and view it in its most verbose form.

### Task 3

<b>Scenario</b>	You are an experienced cook and a vegetarian, you only have about 30 minutes before you have to leave for work.
<b>Instructions</b>	Instructions: Find the list of vegetarian dishes that can be prepared within 20 minutes.

### Task 4

<b>Scenario</b>	You are entering a pie making competition.
<b>Instructions</b>	Bring up a list of all pie recipes.

### Task 5

<b>Scenario</b>	You were a novice user but have now used the website quite a lot. You are browsing a recipe and decide you do not need the step by step instructions anymore.
<b>Instructions</b>	Change the default recipe view.

### Who to use to test?

As described by Deborah Hinderer Sova and Jakob Nielsen in their report “How to Recruit Participants for Usability Studies”[8] the cardinal rule for recruiting is “know thy users”. We have already gathered this information as part of our user-centered design process, to make our personae. To get the most out of our task based evaluation we will need to ensure our 8 test candidates represent a good mixture of our predicted user base. Obviously as students getting other students will be easy, but recruiting computer illiterate testers will prove more difficult. Thankfully, one of our team members was able to get their parents to help out; less than ideal, as that particular user group will be under represented, but probably sufficient.

Nielsen also stresses the importance of focusing on what users do, not just on what they say: they may say they didn’t struggle to find an element on the page, but their eye and mouse movements tell a different story. In the absence of the necessary hardware and software to monitor eye and mouse movements we instead opted to carry out the tests in pairs. This way one conductor can focus on the user and the other on the screen.

However, we can still use what the user has to say about the system, and we used the same rating system as in the CHE. This provides a solid metric with which we can identify key problems with our system.

## 4.2 TBUE Data

The results of the task-based user evaluation are summarised in the following table. As noted, severity ratings are the same as those used for the CHE (Figure 1).

For the sake of brevity, we will not reproduce all of the notes associated with the task-based user evaluation here. However, a couple of the cases were particularly interesting from a usability point of view; what follows is a more in-depth description of the users’ interactions while completing these tasks.

**Task: find vanilla slice recipe and view it in its most verbose form.** The user successfully completed the task by navigating the tutorial and then using the search bar to find the desired recipe. Afterwards he did express some concern over finding the most ‘verbose’ recipe view described by the task. He further elaborated that whilst he was given a summary of all the view types in the tutorial he

### Task 1: Find challenging recipes

User	Problem	Severity
1	Expected filter to apply after pressing enter, but filter button present	1
1	Felt filter button could stand out more, felt it was small but also said it was easy to find as it was where they expected it to be	0
1	Skipped over welcome page tutorial, then wasn't sure which view type to use for experienced cook, cycled through all the views until they found one they liked.	2
2	Read welcome page thoroughly, commented not all information was needed, felt that it slowed them down a bit	1
2	Completed task with no problems, initially drawn to list of recipes but found filters quickly	0

### Task 2: View a Vanilla Slice recipe in its most verbose form

User	Problem	Severity
3	Not obvious what the different views are—descriptions not very obvious, wasn't sure what the most verbose view is, content layout is fine	1
3	Ingredients list not formatting fractions properly, hard to read but can still make out values	1
3	Asked to choose default view, just want to see recipes, slows user down, welcome section too prominent ignore the search box	2
3	Cosmetic issue title not being capitalised	0
3	Search bar not obvious, user scrolled first before choosing to search, but they did find the recipe complained that it took a bit longer	2

### Task 3: Find all vegetarian dishes that can be prepared in less than 20 minutes

User	Problem	Severity
3	Search vegetarian, didn't return anything, filters are there so can recover	2
4	Felt welcome page was not relevant to their task or scenario, slowed them down	1
4	Filter by vegetarian gives nothing, no information to suggest if search return was empty or website errored, assumed website had broken and stopped	4

### Task 4: Find all pie recipes

User	Problem	Severity
5	Pie not left in the search bar, even though title of page says “you searched for pie” inconsistency with pie not displayed in top search bar	1
6	Completed task with no problems	0

### Task 5: Change the default recipe view

User	Problem	Severity
7	No problems reported, went back to page they originally saw	0
8	Nothing on page to show how they could change it, guess its in help, help doesn't help doesn't mention default view. Doesn't know what the default view is or how to change it.	4
8	Stumbled upon answer by going back to home page, and going through first user flow	4

Figure 3: TBUE results table.

failed to decipher which view the task was referring to based on the tutorial text. Perhaps work needs to be done to change the tutorial text to make it more obvious, but the user did confess to not reading the tutorial particularly thoroughly.

The user did state he had no issues with the layout of the tutorial, the text was obvious and easy to read, but it seems the user was more focused on finding the recipe itself; although the tutorial presented relevant information, the user described it as ‘slowing him down’. Further elaboration reveals this user would have been happy to discover the different views on his own, but he did state he probably would want to get this information at a later time. The user gave this a problem rating of 2, but they also claimed a large part of the fault lay in the ambiguity between the task and the website and so their rating may be weighted a bit heavily. From our point of view this purposeful disconnect provided valuable additional information on how the tutorial system is perceived by this user.

**Task: Change the default recipe view** The user remembered that they were given the opportunity to change default view from the tutorial on the home page. They opted to go straight back to the home page and change the default from there. They did however concede that they did not expect the tutorial to show up every time but opted for that route after a quick scan of the recipe page showed no obvious way to change the default.

Another user, however struggled with the task: they spent a long time on the recipe page particularly around the view changing buttons, to no avail. They then attempted to go to help, finding that nothing there was helpful in fulfilling the task. Eventually they stumbled upon the home screen and were presented with the tutorial. They strongly disliked the flow they took and it took a long time to complete the task, so they rated the severity as 4, a very serious problem.

### 4.3 Problems and subsequent changes

The task-based user evaluation highlighted a number of usability issues and bugs in the website. The two highest-severity issues highlighted by the evaluation were:

Regarding search:

Filter by vegetarian gives nothing, no information to suggest if search return was empty or website errored, assumed website had broken and stopped.

and, regarding choosing the default recipe view:

Nothing on page to show how they could change it, guess its in help, help doesnt help doesnt mention default view. Doesnt know what the default view is or how to change it.

Of these, the issue with the filters was a straightforward bug, whereas the issue with default view selection was a serious usability problem. In addition, the mechanics associated with varying presentations of recipes are a core piece of functionality—the ‘killer feature’, in some sense. We will therefore present the issues with and redesign of the functionality around default recipe selection.

In the website prior to the task-based evaluation (hereafter ‘the original’), there was no means to change the default view from the recipe view page (Figure 14). It was possible to change the default view, but only by going all the way back to the entry point of the application and following through the welcome process (Figure 15).

This was extremely surprising to the users, unless they were given the task of changing the default view while still on the welcome page (whereas we would expect users to want to change the default view while browsing and viewing recipes).

To address this issue, we considered the context for changing the default view, and the information needed to do so, taking advantage of our observations of the users during the evaluation.

Changing the view for recipes is strongly associated with actually viewing the recipes, and changing the default view is naturally associated with the rest of the view-changing machinery. In redesigning this functionality, we took account of this context by making the control to change the default view a ‘sibling’ to the existing controls for switching between views.

When changing the default view, the user needs to be able to evaluate the different views and choose one. The key information they need for this is how the view will appear in use. In the redesign, we ensured that the user would have this information by providing a control to make the *current* view the default, rather than, say, a separate choice from a list of alternatives. This allows the user to explore the

site and try out views until they find a presentation they are comfortable with, at which point they can make it the default. This follows a pattern seen in a number of other pieces of software (that is ‘make *this* the default’ rather than ‘*select* a default’), hopefully allowing the user to make use of their previous experience.

In the redesign, we addressed these considerations by adding a new control element underneath the view-changer buttons. When the current view is the default, this is uninteractive, and tells the user that the selected view is the default (Figure 16). When the current view is *not* the default, the control offers the user the opportunity to set the current view as the default (Figure 17).

To further improve the discoverability of this feature, the previously-implemented pop-up help functionality was extended to include an explanation of this feature.

Allowing the user to choose the default view from the recipe page also made it possible to hide the ‘Welcome’ screen, with *its* option to choose the default view, after the user first sees it. This addresses the user frustration, noted above, that it was slightly awkward to get back to the list of recipes.

## 5 Accessibility Evaluation

### 5.1 Validation

**Evaluation methodology** We ensured that the page validated using a combination of the W3C validator[9] and the ‘HTML CodeSniffer’ JavaScript tool[10]. The W3C validator is the accepted standard for validation, and provided us with the most rigorous check; the CodeSniffer tool is quicker and more convenient to use while developing, helping us to avoid creating issues in the first place.

No real ‘flaws’ were uncovered as a result of this process.

### 5.2 Labels and alternatives

**Evaluation methodology** To ensure that all appropriate elements had labels and text alternatives, we used the ‘HTML CodeSniffer’ JavaScript tool; this tool is capable of checking all of the code-level criteria associated with the WCAG 2.0 set of web accessibility guidelines[11], up to level AAA, and will flag up missing labels and text alternatives as errors. It found enough issues that we are reasonably confident that it is accurate.

**Issues found** The design of our website does not make extensive use of images, so there was not much scope for them to cause accessibility issues. However, when checking for image text alternatives, we found that while we had normally added alternative text for meaningful images (such as clickable icons) without needing to pay explicit attention to it, ‘decorative’ images, such as the picture of each finished dish, were left without text alternatives until the audit called attention to it. This would have presented a potentially serious accessibility issue, as the user of a screenreader, for example, would have no way of knowing that those images were meaningless.

For the most part, our initial design had contained labels for all links and form controls, as a natural consequence of the text-oriented design. The only unlabelled control was the text-input box for the site-wide search, which (following a pattern established by many other websites) was mainly described by the immediately adjacent ‘Search’ button. In this case, rather than adding an extra ‘Search’ label, which would only be visually confusing, we added a `title` attribute to the input box to ensure that a screen-reader-user could determine its purpose.

### 5.3 Semantic Markup

**Evaluation Methodology** We evaluated the semantic structure of the markup using a combination of HTML CodeSniffer’s WCAG2.0AAA validation—obviously this can only detect a subset of issues, but while it cannot infer the real semantic meaning of elements it uses heuristics to identify ‘surprising’ markup, which can often indicate areas which should be checked manually—manual inspection of the HTML, and the DOM inspection tools provided by browsers.

**Issues Found** By paying attention to the semantic structure of the document during the design phase, we pre-empted many issues here: the list of instructions is just a heavily-styled `<ol>`, for example.

There are a few areas where the appropriate semantic markup is less obvious: for example, the recipe metadata—category, difficulty, cooking time, etc.—could be presented as a table or just as a list of text; in this case, we erred on the side of simplicity.

There were some fundamentally intractable semantic issues resulting from the general lack of semantic richness in HTML, even after the improvements with HTML5; fixing these accessibility problems is outside the scope of this project.

There were also a few instances where heading elements (`h1`, `h2`, etc.) had been used while prototyping for their visual effect, resulting in an inconsistent outline for some pages. This would have presented accessibility problems to anyone using a screenreader that used the heading structure to help understand the document. This was easily fixed by restyling the logical headings.

## 5.4 Keyboard

### Evaluation Methodology

To ensure that the website was keyboard-navigable (useful not only for power users, but also as an indication of whether the website is likely to be navigable to users with other accessibility issues), we wanted to establish that the following conditions were true:

- Tabbing through elements gives feedback on currently focused element.
- All tab-focusable elements can be activated sensibly by Enter.
- All functionality specified in the requirements can be accessed using the keyboard.

To check the first two properties, we went through each page (counting the search and recipe pages as one page each, rather than expanding all the combinatoric options), tabbing through to ensure that we could see the focus location at all times, and checking that the Enter key behaved as expected.

To check the third property, we went through each functional requirement and attempted to fulfil it using only the keyboard (as we were checking for keyboard-navigability, we used our extensive knowledge of the structure of the site, glossing over temporarily the issue of discoverable keyboard navigation).

### Issues Found

In doing this, we found a number of issues:

**Highlighting of focused elements.** Several interactive elements with more complex visual styling—in particular, the buttons in the header bar and for selecting the level of detail at which to show recipes—had, in the process, lost the builtin visual feedback showing which element was selected for keyboard navigation. This is ironic, as much of the visual styling of these elements existed to provide more feedback for mouse-based navigation.

These issues were generally easy to fix using the `:focus` pseudo-selector to replicate the browser built-in focus feedback (with increased emphasis in some cases to cut through the visual complexity).

**Keyboard activation of extra interactive functionality.** Richer pieces of interaction further from the original hypertext-document model of the web web (in particular, the popup help and tickable instructions) inherited no default keyboard-navigability from the browser, and were for some time impossible to control from the keyboard. This is particularly embarrassing given that the author of this functionality uses a keyboard-oriented browser with an interface borrowed from the UNIX editor `vim` for regular browsing.

These issues were more problematic to resolve, as the `onclick` event (also fired by keyboard activation) is not consistently bound to keyboard activation except in the case of `a`, `button`, and `input` elements. This required a few slightly irritating changes to the markup and styling of the application.



**Tab order of search filters** The search filter form (in our design) sits naturally on the right, both for visual effect and inter-application consistency (many other applications with filterable searches place the filters in the right margin of the results, as a good compromise between being easy to reach and not distracting from the results).

This places the ‘natural’ tab order for search filters *after* all search results: this is very inconvenient—tabbing through a few consistent form elements to reach the results would not be too bad, but tabbing through a large and variable number of search results to reach the filters is very annoying.

Part of the solution to this is obviously to place the filters earlier in the tab order; we did this by placing the filter form before the results in the DOM, as part of our effort to ensure that the site would respond tolerably to small browser windows.

However, this introduces a new issue—tabbing now causes the focus to ‘skip’ across the first few results, which appear visually “before” the filters in a left-to-right reading. This could cause the user to lose track of the focused location.

To resolve this, we added dynamic styling which calls attention to the filter box whenever one of its elements is selected.

## 5.5 Feedforward and Feedback

### Evaluation Methodology

To systematically evaluate our application from a feedforward and feedback point of view, we examined each interactive element on each page, and listed out the visual, dynamic, and textual feedforward cues it provided, the user-visible feedback cues presented on using it, and the real consequences caused by using it.

For each element, we considered these three pieces of data (feedforward cues, feedback cues, and consequences) and arrived at our best determination of whether the element had adequate feedforward and feedback. This was a fairly ad-hoc process done by individuals at various points over the course of the project, although it seemed fairly effective. In retrospect, however, it would have been valuable to explicitly borrow many of the techniques from the CHE—one person compiling the lists of elements, cues, and consequences, and then the group performing a collaborative evaluation directed by those lists rather than by heuristics. But what we did seemed to work reasonably.

Obviously this process is very subjective and rather error-prone. Fortunately, it was a supplement to the other evaluations we had (Collaborative Heuristic and Task-Based); the Task-Based evaluation in particular really exercised the feedback and feedforward properties of our application, (the issues it raised were covered there).

### Issues Found

**Ticking off instructions** One relatively early issue we uncovered was that the cues around the ‘ticking off completed instructions’ steps were wholly inadequate. In the version with this issue, instruction steps were presented as an unadorned list of paragraphs; hovering over each step would cause it to be coloured a dark blue, and clicking on the step would cause it to be styled as ‘strikethrough’ text. This satisfied the functional requirement, but the feedback was potentially unclear, and feedforward was entirely absent.

To address this issue, we styled the list with an empty check-box next to each instruction step—making the suggestion that each step is checkable. Hovering over a step then places a tick in the checkbox, and fades out slightly. Clicking on the step then keeps the tick in the checkbox, and fades the step out further to call attention to the subsequent, unfaded, instruction steps. We feel that this represents a significant improvement in feedforward in particular, as well as making the ticking-off feature more useful by deemphasising completed steps.

There is still some awkwardness integrating feedforward with feedback for this element: after ticking off a step, you have the opportunity to undo that by clicking on the step again. Feedforward for this undo operation suggests that the element should give an indication of what that will do, but this conflicts with providing feedback for the ticking-off operation (as the user will inevitably be hovering immediately after clicking, but the user does not get the best feedback until the cursor is moved away from the element). We could not find a simple mechanism that would provide good feedforward and feedback for these transitions, but despite this, we feel that the major issues with this functionality have been resolved.

**Activation of search filters** Evaluation and testing uncovered some issues with the application of search filters, and the interaction between the top-banner search box and the filtering controls further down the page. It took users some time to be confident as to the effects of various filtering operations.

We addressed the issue of interaction between filters and search box by replicating the search box alongside the filtering controls (scripting was used to ensure that the search terms were identical, to prevent user confusion). This made it easy for the user to understand the interaction between the text query and the filters.

To address the other issues, we altered the filtering controls so that the list of results would update immediately on changing the values of the filters. This gave the users immediate feedback on the effects of their actions, allowing them to become comfortable with the filtering more quickly. It also prevented the user from losing track of which filters they would apply on pressing the ‘Filter’ button.

## References

- [1] “Sass: Syntactically Awesome Style Sheets.” <http://sass-lang.com>.
- [2] “GitHub flow.” <http://scottchacon.com/2011/08/31/github-flow.html>.
- [3] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-computer Interaction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1986.
- [4] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2 ed., Mar. 2007.
- [5] B. Tognazzini, “Togs first principles of interaction.” <http://asktog.com/atc/principles-of-interaction-design/>.
- [6] H. Petrie and L. Buykx, “Collaborative Heuristic Evaluation: improving the effectiveness of heuristic evaluation,” in *UPA2010*, 2010.
- [7] M. McCloskey, “Turn user goals into task scenarios for usability testing.” <http://udel.edu/~rworley/e416/Task%20Scenarios%20for%20Usability%20Testing.pdf>.
- [8] D. H. Sova and J. Nielsen, “How to recruit participants for usability studies.” [http://s3.amazonaws.com/media.nngroup.com/media/reports/free/How\\_To\\_Recruit\\_Participants\\_for\\_Usability\\_Studies.pdf](http://s3.amazonaws.com/media.nngroup.com/media/reports/free/How_To_Recruit_Participants_for_Usability_Studies.pdf).
- [9] “W3C Markup validation service.” <http://validator.w3.org>.
- [10] “HTML\_CodeSniffer.” [http://squizlabs.github.io/HTML\\_CodeSniffer/](http://squizlabs.github.io/HTML_CodeSniffer/).
- [11] “WCAG 2.0.” <http://www.w3.org/TR/WCAG20/>.