University of Cape Town - Department of Computer Science

Department of Computer Science

HONOURS PROJECT REPORT

19 October 2007

Marc Pelteret
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
mpeltere@cs.uct.ac.za
marc@pelteret.co.za

Project Supervisor: Dr Hussein Suleman (Computer Science, UCT)

University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad

ABSTRACT

The Cellphone Shopper project aims to make grocery shopping easier by using technology to aid the process. Key to the project is the management of a shopping list, which is accessible by multiple people through two interfaces: a cellphone and a Web site. This report details the design, implementation and evaluation of the Web interface.

The interface was implemented using AJAX, a Web development technique that uses JavaScript, XML and the DOM to update Web pages without having to reload them. This makes it possible to create highly interactive Web interfaces. The Yahoo! User Library, an AJAX framework that provides numerous interface elements and tools, was used in constructing the site.

The interface was evaluated through user testing and heuristic evaluation. The evaluators were happy with the interface on the whole. The majority liked its aesthetics and thought that, on a whole, it functioned well. However, the evaluations uncovered several usability issues. The most serious problems are that the interface does not provide feedback to the user when it is communicating with the server, and almost all users failed to understand and use the store layout design page. However, these problems are rectifiable.

Despite its flaws, the system has the potential to be a useful tool for shoppers and there is a wealth of possible features that can be added to it in the future.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – Evaluation/ methodology, Graphical user interfaces (GUI), Interaction styles, Prototyping, User-centered design; H.3.5 [Information Storage and Retrieval]: Online Information Services – Web-based services, Commercial services.

General Terms

Design, Human Factors, Management

Keywords

Shopping, cellphone, AJAX, interface, grocery

ACKNOWLEDGEMENTS

A special thank you to the following people for being willing to sacrifice some of their time to help design and evaluate my project:

- Waheeda Amien
- John de Villiers
- Kerryn Evans
- Bertus Labuschagne
- Christopher Parker
- Patricia Pearson
- Ilda Ladeira
- Andrew Maunder
- Tammy Martin
- Liyaqat Mugjenkar
- Alfred Mukudu
- Brenton Tenner
- Cara Winterbottom
- Chimbala Wright
- Sebastian Wyngaard

An extra special thank you to:

- Bertus and Chris for being involved in so much of the project.
- Mr Dudley Smit of Checkers, Rondebosch for allowing me to take photographs of the store.
- Tony and Avril Morris and Renisha Deolal of Anmore Distributors for printing and binding my project, and being so accommodating and helpful.
- Hussein Suleman for proposing such an interesting project, and for all the interesting discussions during the year.

Finally, the biggest thank you of all goes to my family - my brothers, Jean-Paul and Eugene, and especially my parents, Robin and Denise – for all their support and encouragement during my stint at university. I could not have got through it without you!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. PROBLEM OUTLINE

Grocery shopping can be a nightmare, especially when one person decides what needs to be bought but another person has to do the buying. Typical problems include:

- Difficulty in sharing the shopping list – what is the list written on and where is the list kept?
- One person adding something to the list and another wondering who added it and why.
- Going shopping, only to realise that the shopping list has not been brought along.
- The buyer not knowing which brand of item to buy.
- Co-ordination: who does the shopping and when?
- When the buyer gets to the store, they do not know where items can be found and spend a long time wandering around the store looking for them. Also, shoppers may visit the same section several times because they do not realise that several of the items on their lists can be found in that section.

The key aim of the Cellphone Shopper project is to make grocery shopping easier by using technology to aid the process. At its core is the management of a shopping list stored on a central server which is accessible by multiple people through two interfaces: a cellphone and a Web site.

The typical use case is that of one user creating the shopping list for the current day or week and another person viewing that list on a cellphone while they are in the store doing the shopping.

1.2. OVERVIEW OF THE SYSTEM

Cellphone Shopper has three major components:

- A back-end, which consists of a Web application and a database. (The task of implementing this was allocated to Graham Hunter).
- A Web interface. (This was allocated to the author).
- A cellphone application. (Allocated to Tshifhiwa Ramuhaheli).

The back-end is responsible for the storage of data used by the system. This data is made accessible to the Web and cellphone interfaces through a SOAP-based API. The back-end also performs any calculations needed by the system.

The Web and cellphone interfaces allow users to view and manipulate their lists and various other features of the system. In addition to this, the Web interface is the starting point for new users, who must use it to sign up for the service and create and set up their lists.

Figure 1 shows how these components connect together.

The back-end's Web application runs on an Apache Tomcat server and accesses the MySQL database. The Web interface is served on an Apache HTTP server and interacts with the back-end using AJAX. The cellphone application is programmed in J2ME and communicates with the back-end over GPRS.

1.3. PLAN OF DEVELOPMENT

This report outlines the work the author did on the Web interface.

It begins with some background to the project, looking at systems related to it and technologies that could have been and were used in creating it.

Following that, the design of the interface application and it's implementation are looked at. The next chapter then details how it was evaluated and what the results were.

Finally, the report is concluded and possible future work is listed.

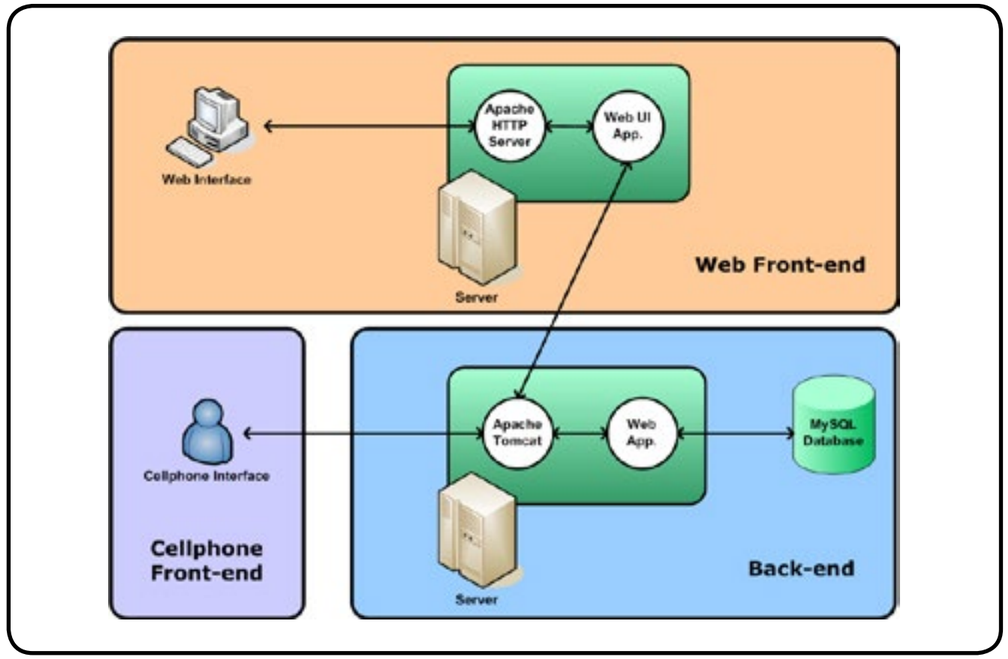


Figure 1 : A diagrammatic view of the system

CHAPTER 2

BACKGROUND

This chapter begins by looking at related systems, specifically two shopping systems and three Web sites that maintain lists and offer services based on those lists.

The aim is to create an interface that is interactive and simple and quick to use. To do this, the AJAX Web development technique was be used. In the Section 2.2, AJAX will be examined more thoroughly and examples of its uses given. A selection of AJAX frameworks also will be examined.

Linked to AJAX is the topic of efficient communication, as it directly affects interactivity. This will be discussed in Section 2.3.

2.1. RELATED SYSTEMS

In this section, two types of systems will be examined: shopping systems and Web sites that manage lists and offer services based on them. They will be compared to Cellphone Shopper to see if they have any features that could be used in the project.

2.1.1. Shopping Systems

Two shopping systems will be looked at: a mobile shopping assistant, developed by Wu and Natchetoi [36]; and Safeway UK and IBM's Easi-Order [4].

The mobile shopping assistant is a J2ME application that runs on a cellphone. It allows a customer to access Web Services published by a store while they are in that store. Example services include access to product descriptions and promotions and the ability to locate products within the store.

The customer begins their shopping by creating a shopping list of products or product categories. To help the customer find an item, the assistant displays a floor plan showing the user's current location, the location of the product and the shortest route.

Easi-Order is a PDA-based application that aims to extend Safeway UK's Collect & Go system, an ordering service that allows customers to place orders remotely via phone or fax and pick up their bagged groceries at the store the following day. Easi-Order extends this service by allowing customers to use a PDA to place orders.

Central to the system is a personalised shopping list, which is a list of items a customer has previously bought. The system also lists items on special, items that customers can buy with Safeway loyalty points and products it recommends based on the customers' shopping habits. From these the customer draws up a list of items they want currently. They also can request items not on these lists.

Like the mobile shopping assistant, Cellphone Shopper also is a J2ME application and also displays floor plans and shopping routes. However, it is not a store-based system – it is accessible anywhere. And while Cellphone Shopper provides a history of previous shopping lists, this history is not the focal point as it is with Easi-Order.

2.1.2. List Management and Services

Three Web sites will be examined in order to see what lists they maintain, how these lists are managed and what services are offered for or based on these lists.

Amazon.com [2] maintains some lists for a specific user and offers several services based on these lists. Here are just some of the lists it maintains:

- Recently Viewed Items
- Open & Recently Shipped Orders
- Shopping Cart
- Wish List
- Gift List
- Shopping List
- Wedding Registry
- Amazon Friends
- Interesting People
- Reminders
- Listmania! Lists

Except for the first two lists, all of these are user managed. Listmania! is a relatively new feature. It allows users to maintain lists of products they find interesting and share them with others. You can view a list and select items from it to add to your Shopping Cart or Wish List.

One service that Amazon.com offers is that of recommendations, where it recommends products that you might be interested in buying. The products are chosen by examining the items you have purchased, items you have told Amazon.com you own and items you have rated, and then comparing your activity on the site with that of other customers. The list of recommended items may change daily and can be quite extensive. Again, you can choose to add items to your Shopping Cart or to your Wish List.

Facebook [8] is based on lists. Users maintain a list of their friends and interests and the system tracks changes and additions to these lists, sharing these changes with the user's friends through what it calls a "news feed". This way, users can track what their friends are doing – what is new in their lives, what they like and so on.

The Woolworths online shopping store [35] allows users to maintain lists of products that they may want to buy regularly. The aim is to save users the tedious task of having to add the same items to their "shopping bag" every time. Instead, you can simply select that list and the items are added for you, and you can continue to add other items to the bag before checking out.

Cellphone Shopper provides a "news feed"-type feature. It informs users of recent changes to the list and other news. The system also provides a regularly-bought items list, the aim of which is to save users time when they are compiling their shopping lists.

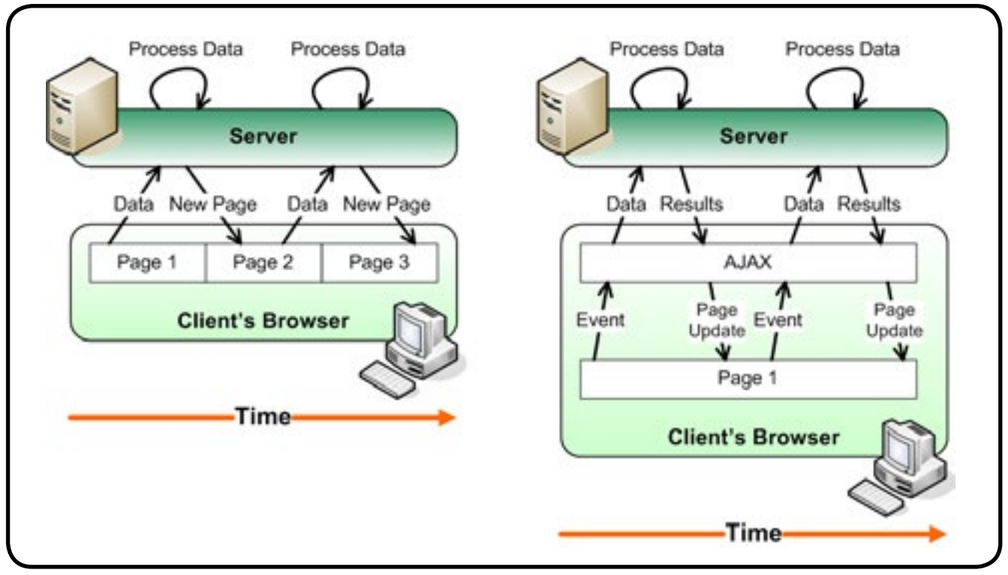


Figure 2 : The classic Web application model vs the AJAX application model

2.2. ASYNCHRONOUS JAVASCRIPT AND XML (AJAX)

AJAX is not a technology; it is, in fact, several technologies that have been grouped together to form an approach to Web application development. As Garrett [11] explains, it incorporates:

- standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- and JavaScript binding everything together.

Classic Web applications are based on a multi-page interface model [11, 24]. In this model (Figure 2, left), a user action on a Web page triggers an HTTP request, which is sent back to the server. The server performs some actions based on this request and then returns a Web page to the client, which replaces the original page with the one it was sent.

AJAX uses a different model (Figure 2, right), namely that of a single-page interface [11, 24]. When the user triggers an event that leads to communication with the server, rather than re-loading the entire page to display the results of that interaction, changes are made to elements *within* the page. In this way the user's interaction with the Web application is asynchronous – it is independent of communication with the server.

The core reason for AJAX coming into being is interactivity, which is closely associated with usability [9]. Teo et al. [32] investigated the effects of the level of interactivity of a Web site on a user's attitude towards the site. Their results suggest that an increased level of interactivity has positive effects on a user's perceived satisfaction, effectiveness, efficiency, value and overall attitude towards a Web site.

Directly connected to interactivity is responsiveness, in particular, user-perceived latency. User-perceived latency is the period between the moment a user issues a request and the first indication of a response from the system [24]. It is desirable for a Web application to be as responsive as possible.

There are two primary ways in which user-perceived latency can be decreased: (1) decrease the round-trip time and (2) allow the user to interact with the system asynchronously. AJAX does both.

The core part of AJAX that allows communication without the need for the Web page to be completely refreshed is the `XMLHttpRequest` element [31]. It allows for a *delta-communication* style of interaction [24], where only state changes are communicated between the client and server, rather than HTTP requests with full-page responses. Hence the round-trip time of client-server interaction is decreased.

As mentioned above, the user's interaction with an AJAX Web application is asynchronous: the client browser can make requests to the server without making the user wait before they can interact with the application again. In addition to this, the use of JavaScript, in particular, means that many of the application's responses can be handled on the client side without involving the server – for example, form validation and various forms of data editing.

There are two issues worth noting with AJAX [24]. The first is that the initial download of the AJAX code, often referred to as the *engine*, introduces some latency for the user. However, data transfers (while using the system) are smaller and compensate for this. Second, it is possible to burden the client by over-using client-side functionality.

2.2.1. Example Web Sites

There are many Web sites that make use of AJAX. The following examples illustrate the wide variety of uses for it.

Google Suggest [15] – as you type into the search box, the system offers suggestions in real-time on how to complete what you are typing. Many other Google sites make use of AJAX, including Google Maps [14] (its pan and zoom functions adjust your view using AJAX), Google Mail [13] (which provides, among other things, real-time chat using AJAX) and Google Calendar [12] (which uses AJAX extensively – for example, to provide the user with the ability to drag events from one timeslot to another).

Live Search [23] – the image search of Microsoft's Live Search makes extensive use of AJAX. The results of a search are returned as thumbnails. The number of thumbnails initially returned depends on the size of the screen on which the Web page is being viewed. As the user scrolls down, more images are dynamically added to the page. This way all the returned images can be shown on one page, but the user does not wait for all the thumbnails to download – they are downloaded only when the user scrolls to them. When the user hovers over a thumbnail, a window pops up and displays a slightly enlarged version of the image as well as details such as its filename, dimensions, file size and URL. Finally, the site has a "Scratchpad" that allows the user to keep track of particular images by dragging-and-dropping images from the search results on to it.

Panic Goods [29] – this Web site sells t-shirts. You select the t-shirts you like by dragging them to your shopping cart at the bottom of the page.

LiveMarks [22] – this site allows you to watch bookmarks in real-time as they are added to del.icio.us, a social bookmark Web site where users can maintain their bookmarks and share them with others.

AjaxTrans [1] – this system translates from one language to another as you type, sentence by sentence.

2.2.2. Frameworks

Web development is a complex task. Developers have to deal with issues such as content presentation, loading times, aesthetics, navigation, interactivity and security [6]. One of the biggest problems is browser compatibility, which affects, among other things, the way CSS displays and JavaScript operates. Quite often, developers have to use non-standard tricks – "hacks" – to get around this issue, or they simply have to accept the differences.

Frameworks aim to simplify the task of development. More specifically, the goals of these frameworks are to [24]:

- hide the complexity of developing AJAX applications – which is a tedious, difficult, and error-prone task,
- hide the incompatibilities between different Web browsers and platforms,
- hide the client/server communication complexities, and
- achieve rich interactivity and portability for end users, and ease of development for developers.

To achieve these goals, the frameworks provide a library of user interface components and a development environment to create re-usable custom components.

There are numerous AJAX frameworks: AJAX Patterns [10] lists more than 200 of them at the time of writing. Because there are so many, only a small selection will be looked at here. Specifically, the following frameworks listed in Wayner and Turner, and Wang [34, 33]: Direct Web Reporting, Dojo, Google Web Toolkit, Microsoft Atlas, Open Rico and Prototype, Prototype and Scriptaculous, Yahoo! User Interface Library and Zimbra's Kabuki AJAX Toolkit.

A framework that is pure JavaScript may be best for the project, so Direct Web Reporting and Google Web Toolkit, which are both Java-based, are not appropriate. Microsoft Atlas is also not appropriate because it is deeply integrated with Microsoft's .NET environment and Cellphone Shopper is not run in a solely Microsoft environment.

While all the remaining frameworks are adequate, it was decided that the Yahoo! User Interface Library is the best choice. Dojo has a broad collection of widgets, but its documentation has gaps in it [34]. Zimbra's collection of widgets is basic and its documentation is also lacking. The combination of Open Rico and Prototype lacks some pragmatic tools, such as a tree. Scriptaculous also lacks complete documentation. The Yahoo! Library has good documentation (that contains many examples and a great deal of code), contains many of the standard tools (such as an animation library and a tree collection) and is easy to work with

[34].

It is possible to use multiple libraries, either in part or in whole. So should the chosen framework lack something that another has, it can be augmented.

AJAX was an appropriate approach to take for creating the Cellphone Shopper Web interface. It offers interactivity and speed, two elements that are important for a service that will be used on a regular basis. The availability of various GUI "widgets" in the Yahoo! framework facilitated the building of an interface that is easy for the novice user and quick for the experienced user.

Linked to the speed of an AJAX approach is the efficiency of the underlying communication protocol. This will be examined in the next section.

2.3. EFFICIENT COMMUNICATION

Efficient communication is important for a Web application, particularly an AJAX one. In this section, four light-weight communication protocols will be looked at: XML-RPC, SOAP, REST and JSON-RPC.

XML-RPC is remote procedure calling using HTTP as the transport and XML as the encoding [37]. It is designed to be simple while allowing complex data structures to be transmitted.

SOAP is intended for exchanging structured information in a decentralized, distributed environment [16]. Its key aims are simplicity and extensibility. SOAP messages are encoded in XML and can be transmitted over a variety of underlying transport protocols.

SOAP is the successor to XML-RPC. It works with objects rather than remote procedure calls, but has a greater overhead than XML-RPC [21].

Both XML-RPC and SOAP are widely supported, having been implemented in many languages for many operating systems and environments, both open source and commercial.

Representational State Transfer (REST) is an architectural style, not a protocol. However, it is very often referred to in protocol discussions [38, 25, 30]. In this context, it is used loosely and refers to the sending of data over HTTP without the use of a particular message encoding (you can use any, including an XML-based format) [17].

JSON-RPC is an RPC based on JavaScript Object Notation (JSON), which is used to encode its messages [3]. HTTP is used to transmit the messages. JSON is data-orientated, is based on a subset of JavaScript and is easy for humans to read and write [18]. It is also simpler and smaller than XML [20].

JSON is (a) a native format for JavaScript, which AJAX uses, and (b) smaller than XML. However, it is not as widely supported as XML. While this lack of support is not a problem for a Web interface, particularly one based on AJAX, it is a problem for other interfaces, such as a cellphone. XML has far wider support.

The online discussion on XML-RPC vs SOAP vs REST has been a long-running one that appears to have no definitive winner [38, 25, 30]. It appears that SOAP is somewhat verbose, but XML-RPC may be no better. Some people argue that XML-RPC has widespread support because it has existed for a number of years, but SOAP appears to be just as widely supported. Many think that a REST approach is enough (using XML, or maybe even SOAP, for encoding the message).

With no definitive best-choice, it is up to the developers of a particular project to decide what they want to implement. It is possible to support multiple protocols, so it is not the case that one has to be chosen over all the others.

2.4. SUMMARY

This chapter looked at the techniques and technologies that were used in developing the Web end-user interface for the Cellphone Shopper system.

Studying some shopping systems and Web sites that maintain lists provided ideas for features that could be implemented in the project to aid and inform users. In particular, the system includes "news feed" and a list of regularly bought ("favourite") items.

The Web development technique known as AJAX was used in the Web interface. The technique's core aims of interactivity and speed are important to Cellphone Shopper. The Yahoo! User Interface Library was chosen from of the above selection of AJAX frameworks for use in the project. It was selected because of its extensive documentation and numerous examples.

Linked to AJAX is the topic of efficient communication. Four protocols were examined: XML-RPC, SOAP, REST and JSON-RPC. The project team decided to use XML, given that it offers structure and there XML parsers available on the majority of programming platforms. It is possible to include support for the other protocols at a later stage, if required.

CHAPTER 3

DESIGN AND IMPLEMENTATION

This chapter looks at how the Web interface was designed and then implemented.

It begins by listing the interface's final features and then briefly discusses the technology and tools used to create the interface.

The design involved a wide range of people and consisted of three phases, which are discussed in sections 3.3 to 3.6.

Finally, section 3.6 describes the protocol used to communicate with the back-end.

3.1. WEB INTERFACE FEATURES

This section explains the major features of the system, looking at each page in the site.

3.1.1. Login (Index), Registration and News Pages

The client-side login system been implemented entirely in AJAX. It uses the server to verify the user's password, but otherwise manages the user's login itself. It uses cookies to track when a user is logged in and which user they are. Only if they are logged in can they access any of the core Web pages. If they try to access any of the pages directly, the system re-directs them to the login page.

The user's password is encrypted using the MD5 algorithm. This hash is what is sent to and stored on the server, so only the user knows their password (the system administrators, for instance, cannot recover it).

If the user has not registered, they may choose to do so by clicking on the "register" link that is on the login page. This will take them to Registration page, which has a form where they specify their various details and select a unique user name. The system searches through its database to ensure that the user's chosen user name has not been selected and informs the user if it has. The registration process will not complete unless the entire form has been completed and a unique user name has been entered.

The first time a user logs in they will be directed to the My Preferences page (described below). Subsequent logins will direct them to the My News page. This presents them with a list of news items that tells them what lists have been modified since their last log in (and by whom) and what reminders (described below) are due.

Figure 3 shows a UML activity diagram of the login system.

3.1.2. My Preferences Page

On this page (Figure 4), users control configuration options that are particular to them. They can change their various details (name, user name, e-mail address, cellphone number and password) and choose their "favourite shops" and "trusted users".

Favourite shops are shops where the user usually buys goods from. These shops can be associated with a list or item. Most people will buy from only a few shops, so rather than having them search through all the shops in the system (which could be hundreds in a full system available to the whole country) each time they add a list or item, they select only those they are most likely to use. So when they add a list or item, they select from a much shorter list, thus saving them time.

Trusted users are other users that the user might permit to access their lists. The aim of this feature is the same that of favourite shops: to save the user time by shortening their list of choices when adding a list.

3.1.3. My Lists Page

Users can have more than one shopping list. The My Lists page (Figure 5) allows them to manage their lists. They can add and delete lists, as well as edit each list's details (the list's name and the default favourite shop associated

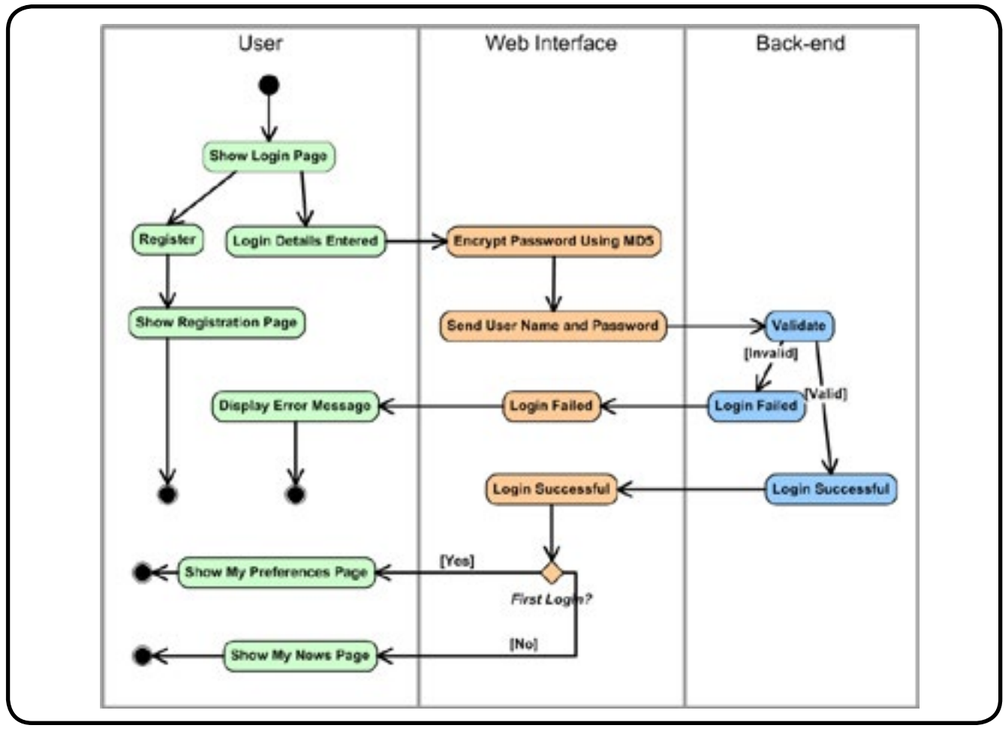


Figure 3 : An activity diagram of the login system

Figure 4 : The My Preferences page

with it).

Lists have access rights. The owner of a list can choose which trusted users may access their list and may assign rights to those users. There are three rights: view the items in the list, add items to the list and delete items from the list. Any user given access to the list can view it, but the other two rights must be assigned by the owner.

Each user has a number of special lists. The aim is to save the user time: instead of manually entering a new item that they have added before, they can simply select it from one of the special lists. These lists are:

- Favourites – items that the user often adds. The user maintains this list.
- Previous items – a system-generated history of all items the user has previously added.
- Deleted items – any items the user has added and then removed during their current session.

3.1.4. List Editor Page

This page (Figure 6) allows the user to manage the items in a list. They can add, delete and edit items.

Items can be marked in various ways:

- Each item can be assigned a category to which it belongs and a shop from which it should be bought.
- An item can be flagged as being private, so that only the user who added it and the list owner can see it. The user who added the item may not want other users with access to the list to be able to see it, either because it is private by nature or for some other reason (e.g., the item might be a gift for someone else who can view the list).
- A user might be unsure as to whether or not an item is needed, so they can tag an item as "uncertain" and someone else who has access to the list can decide whether or not it is needed.

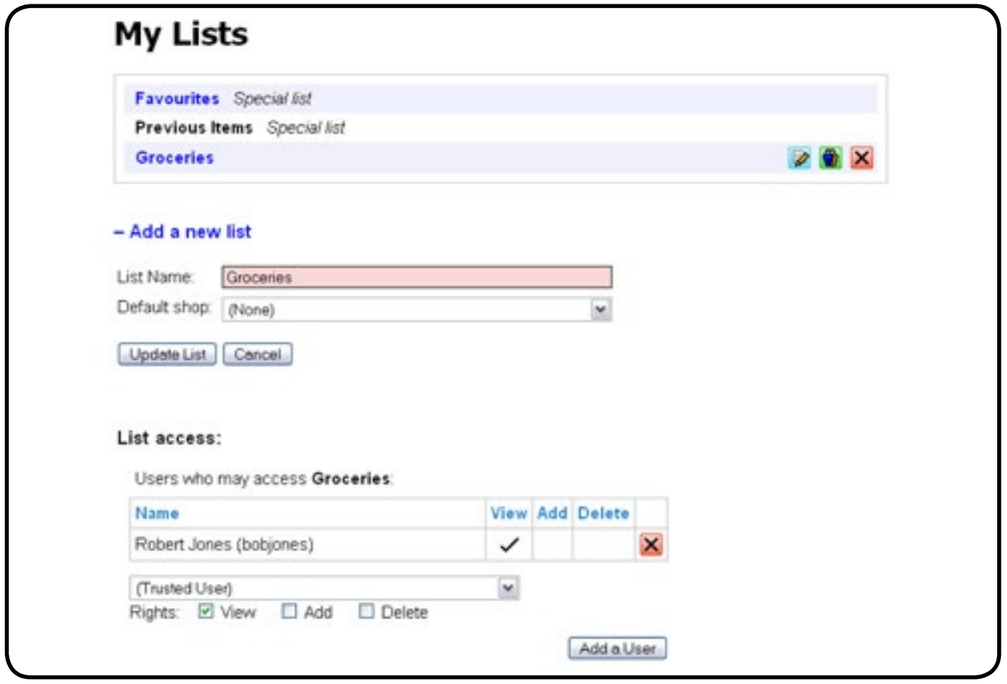


Figure 5 : The My Lists page

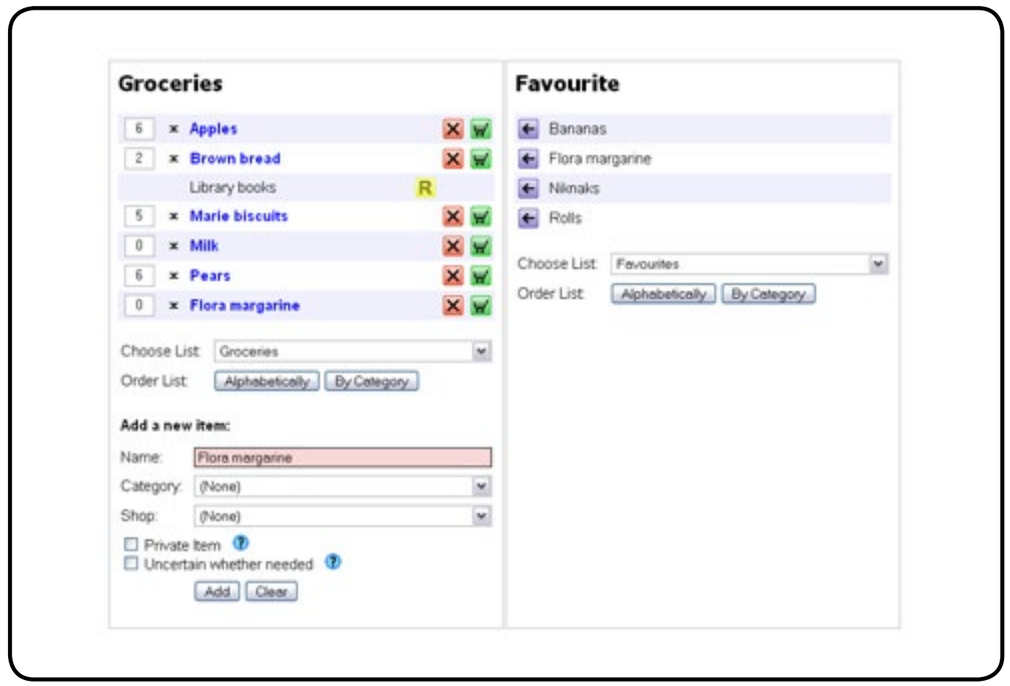


Figure 6 : The list editor page

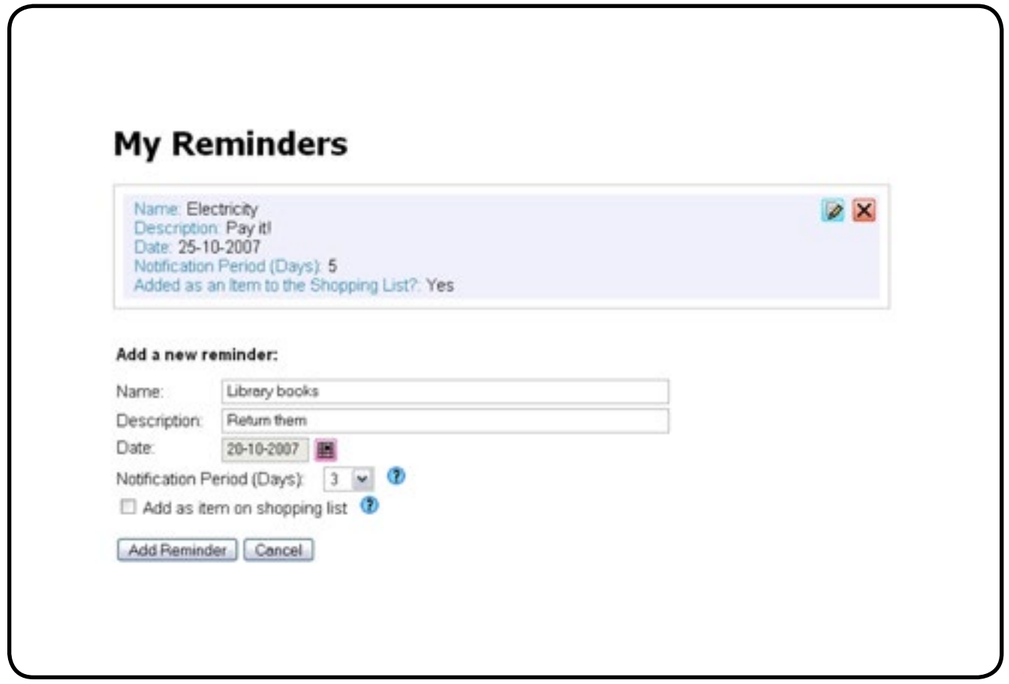


Figure 7 : The My Reminders page

Each item can have a note included with it. Items can also be checked off as they are bought. When this happens, they are moved off the current list and into the item history.

Lists can be viewed in two ways while editing: alphabetically and by the categories to which the items belong.

3.1.5. My Reminders Page

Users can set reminders on this page (Figure 7). The idea is for users to be able to have the system remind them about tasks while shopping. For example, Pick 'n Pay allows customers to pay utility bills at its stores, so the user could set a reminder to settle the electricity bill while buying groceries.

Reminders can also be deleted and their details edited after they have been added.

Each reminder has a "notification period": the number of days of prior warning the system gives the user before the reminder is due. Once the date of the reminder has passed, it is removed from the system.

There is also an option to show the reminder on the user's shopping lists as an item. This is primarily to allow someone viewing the shopping list on a cellphone to see the reminder without having to go out of the list and into a separate screen.

3.1.6. My Shop Layouts Page

A shop's layout is used to order a list's items in by section while using the list on the cellphone. The aim is to help users locate items within the store. On this page (Figure 8), users can view, add and edit shop layouts. A shop layout consists of a number of aisles, each of which contains a number of item categories.

3.1.7. Help

Tooltips are used frequently throughout the system, so that when a user hovers the mouse cursor over a button (for example) they are told what the button does. Figure 9 shows an example of this.

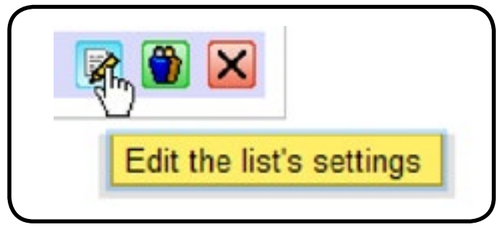


Figure 9 : A tooltip explaining the "edit list's details" button of the My Lists page.

3.2. TECHNOLOGY AND TOOLS

The Web interface is a client-side application created using HTML, Cascading Style Sheets (CSS), JavaScript and the Document Object Model (DOM) built into all browsers. It is a combination of hand-written code and functionality provided by the Yahoo! User Interface Library.

A prototype was built to test the technologies (Figure 10). The lists are obtained from a stub PHP script which is meant to simulate the server. It generates a list of items based on which list (the main list, list A or list B) is requested and returns that list as an XML document. The list is then parsed on the client side by the application and displayed. The main list can be manipulated: items can be removed from it, new items can be created and added to it, and items from lists A and B can be added to it.

The prototype is very simple, but was intended only to check that the technologies are adequate for the project and to get an idea of how they work. It also served to demonstrate an idea that wouldn't work. The initial plan was to make use of the drag-and-drop functionality provided by the Yahoo! toolkit. However, while creating the prototype it was discovered that the interface (Figure 11) became very slow when the drag-and-drop list contained a large number of items.

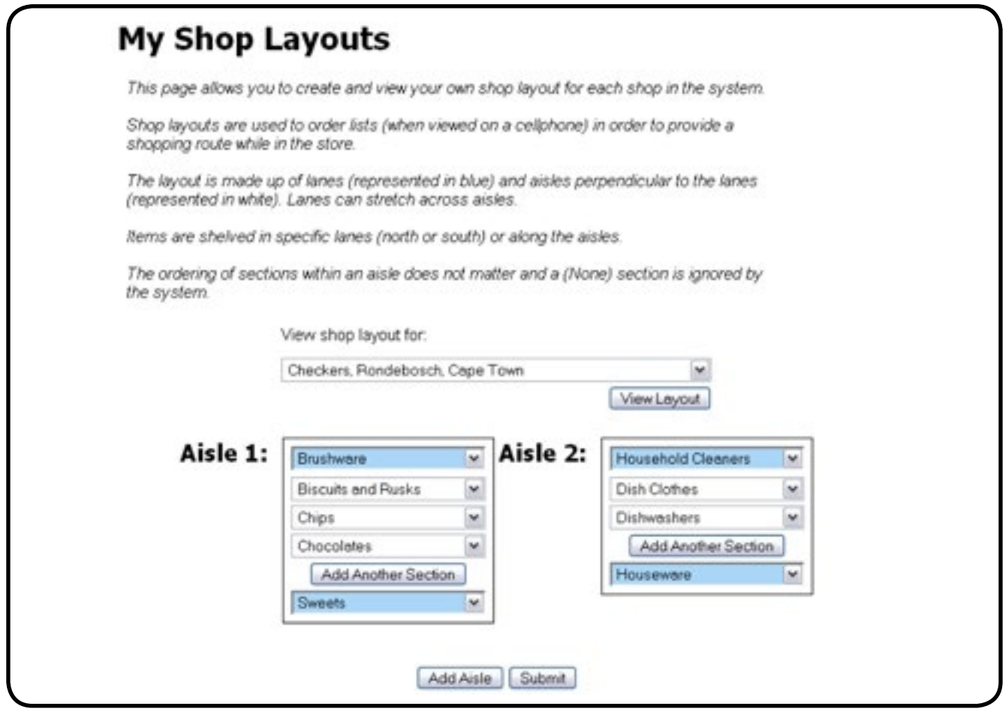


Figure 8 : The Shop Layouts page

Figure 10 : The technology prototype

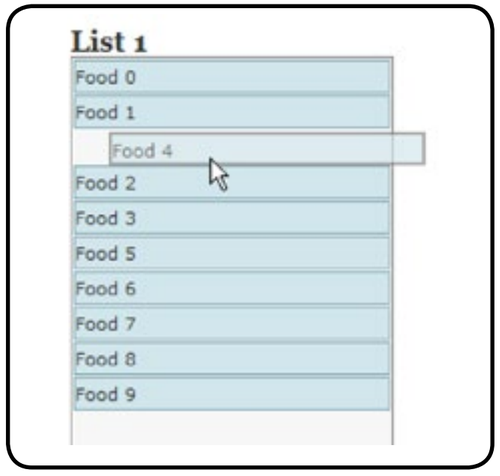


Figure 11 : The prototype drag-and-drop interface

3.3. DESIGN INTERVIEWS 1: FEATURES

The above-mentioned features were obtained through a combination of project team brainstorming and design interviews involving potential users and usability experts. This section and the next two detail the design interviews.

Cellphone Shopper is aimed at anyone who uses a shopping list, whether they are in charge of it or simply use it in some fashion. This means that the system needed to be designed to be used by a wide range of users from many different classes – young and old, parents and children, comfortable with technology or not, and so on.

User-centred design is the key to designing such a system. Total user experience drives the design [26] – i.e., users and their tasks are what is focused on. Users are involved in all stages of the product's development and continuous user input ensures that the team has a good understanding of what users do and want to do, and how well the design satisfies those tasks.

The first stage of the process was user requirement gathering. Users were interviewed to determine what they required and wanted to see in a system like Cellphone Shopper. A typical interview proceeded as follows:

- The interviewees were questioned on their current shopping behaviour and how they presently manage their shopping list.
- The proposed features for Cellphone Shopper were explained and questions were asked about how the interviewees would use these features.
- To round off, interviewees were asked for their opinion on the proposed system and they were asked to suggest any other features they would like to see in it.

The interviews provided many pieces of useful information:

- People are interested in having a system like Cellphone Shopper. The primary attraction is the ability to use a cellphone to manage a list because of the convenient access. For instance, one user expressed her frustration at arriving a shop and realising that she had either left her list at home or in the car. She never, however, forgets her cellphone.
- Most of those who were interviewed shop at more than one store.
- One user has two lists: one for groceries, the other for non-grocery items.
- The ability to order the shopping list by section while using the list on the cellphone is a feature that all the interviewees found interesting and felt would be very useful. The idea is for the shopping list to be ordered such that the shopper can move through the store section by section, starting from the store's entrance. One person supported this by saying that they found it irritating that at times they have to return to a section they have already visited because they neglected to get an item while they were in the section. One interviewee, however, works a little differently: instead of moving through the store from the entrance to the exit, he moves through it by how he stacks items in his trolley. So he would like to be able to order the sections for himself.
- In general, those interviewed do not want the system to automatically add items they regularly buy – they would prefer to have full control. The reason, for one person, is that they could end up buying an item when they do not need it, because the system would not know that they still have some of that item left.

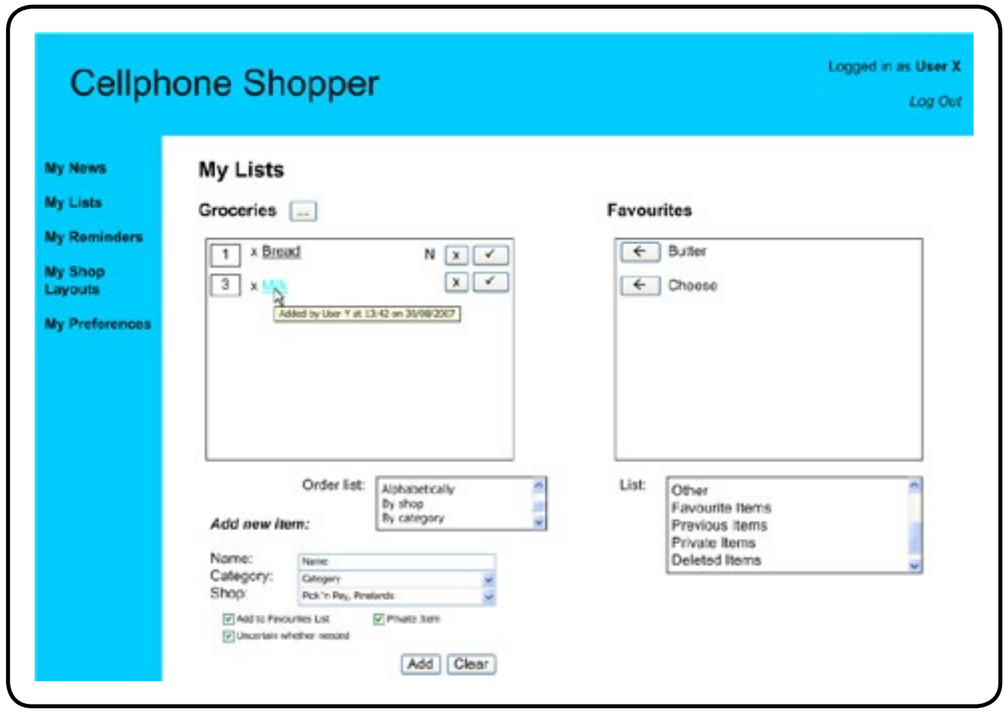


Figure 12 : A page from the low fidelity prototype used in the second design interview

3.4. DESIGN INTERVIEWS 2: INTERFACE PROTOTYPE

After interviewing people who use shopping lists and drawing up a list of features that would be included in the system, the cell-phone and Web interfaces were constructed.

In order to test the interface designs, a focus group was organised. The group consisted of four volunteer Computer Science students, two from Masters and two from Honours, each with interaction design train-ing. The idea was to have a group of "ex-perts" scrutinise and critique the interfaces.

The designs the group was shown were low fidelity paper prototypes which were drawn in Microsoft Visio (Figure 12 shows an example page). The group was walked through each

screen of each interface and given the chance to express views on each screen's design and offer any ideas or advice they could think of.

Like the shopping list user interviews, the session was very constructive. Again, a number of new ideas were offered. Some of these ideas were implemented in the system, while others have become suggestions listed under Future Work (Chapter 6).

The following are two major ideas the group suggested that were implemented:

- Trusted users, who can be added via an auto-completed text box (i.e., rather than one user having to type in the full name or e-mail address of another to add them as a trusted user, they can simply start typing in those details and the system will suggest how to complete what they are writing). Figure 13 shows the final interface's auto-completed text box.

Figure 13 : The trusted users auto-complete text box implemented in the final interface

- The use of icons, which were not pre-sent in the prototypes. The group as a whole strongly believed that they should be used. The group also recommended that common icons be used in both in-terfaces – for instance, a trolley icon for the interface element that the user uses to mark an item as bought (Figure 14).

Figure 14 : The "mark item as bought" button of the final interface, showing the trolley icon

3.5. DESIGN INTERVIEWS 3: INTERFACE IMPLEMENTATION

The cellphone and Web interfaces were con-structed over the three weeks following the second design interview. A third design in-terview was then organised to evaluate the results. Again, four "experts" evaluated the system: the two Masters students involved in the second design interview and two Hon-ours students (not the two who originally participated; they were unfortunately unavail-able).

The Web interface that was presented was an interactive "shell". Most of the core pages were implemented and their major in-terface elements were present and could be interacted with. For instance, on the My Reminders screen the user could use a form to add a reminder and its details would be listed on the screen; this reminder could then be edited and its details would be updated on the screen immediately. The site had no back-end connectivity, though it did call a stub PHP script for two example lists.

The group made several constructive criticisms and offered several useful sugges-tions:

- The form of the My Preferences page "grows" when the user wants to add an-other shop – in Figure 15, the fourth row of favourite shop combo boxes was added when the "Add Another Shop" button was pressed and the fourth text box of the trusted users was added when "Add Another User" was pressed. It was suggested that instead of doing this, one should rather have one form ele-ment for each section and above it show a list of the shops or users the user has already added. This is what was done in the final implementation (see Figure 4).
- The edit icon on the My Reminders and My Lists pages should be changed be-cause it isn't clear showing what the

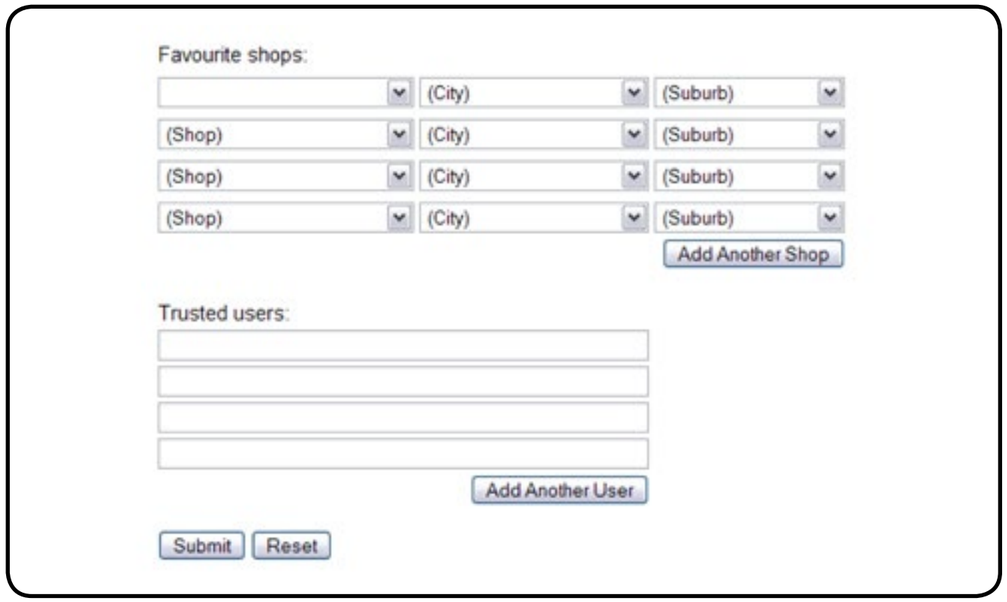


Figure 15 : Part of the My Preferences page of the interface implementation

but-ton is for. Figure 16 shows the original button and what it was changed to in the final interface.

- The "Add as item on shopping list" option on the My Reminders page should be select by default. The group felt that reminders would be more useful in gen-eral if they appeared on the shopping list..
- In the various lists, the row that the mouse cursor is over could be highlighted to help show the user which item in the list they are manipulating with the various buttons on the right (Figure 17).

One of the group was concerned about the fact that as a shopping list grows, the "add new item" form is pushed off the screen. They suggested that an inline frame (which displays one Web page inside an-other) be used for the list. However, as someone else in the group pointed out, the problem is that there is not enough screen space anyway. The result of using an inline frame would be that only a few items would appear on the list and the user would be forced to scroll anyway – i.e., the choice is between making the user scroll an inline frame or scroll the browser window. With that being the case, the group felt it was best to leave things as they are and not use an inline frame.

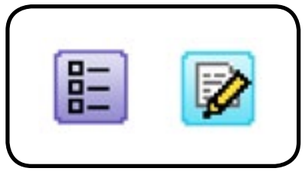


Figure 16 : The "edit" button original and that of the final interface

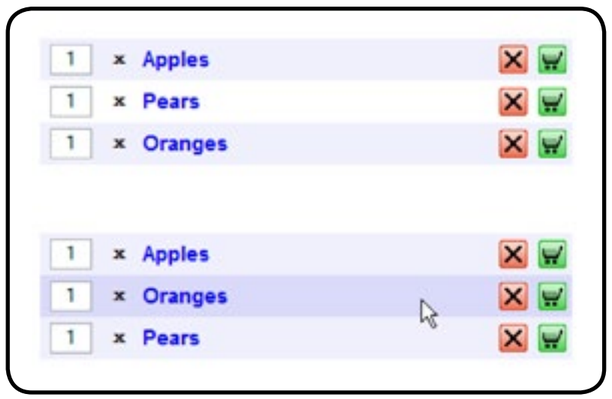


Figure 17 : As the mouse is hovered over a list row in the final interface, that row is highlighted

3.6. BACK-END COMMUNICATION PROTOCOL

3.6.1. Protocol Encoding

The back-end is a Web Service programmed in Java and provides a REST interface, which the Web interface uses to communicate with over HTTP. Data are retrieved using GET requests and sent using POST. Parameters are URL-encoded for both methods.

The responses are encoded in a custom XML data format. The aim of the custom format is to keep the data structured so that it can be parsed by an XML parser, some-thing which is available for most mobile- and computer-based programming languages, but make the structure as lean as possible so that messages do not become bloated.

So, for example, instead of encoding a shopping list item as is done in XML listing 1 in Figure 18, it is encoded using the format of XML listing 2. While the differences may seem trivial, the second list (which has 172 characters, including whitespace) is approxi-mately 20% smaller than the original (which has 214 characters).

Streamlining the XML response is important for the cellphone in particular, due to the general lack of broadband connectivity.

The Web interface uses the Yahoo! User Interface library Connection Manager utility for communicating with the server. This utility provides a simplified interface to the Web browser's `XMLHttpRequest` object.

3.6.2. Protocol Semantics

The API for communicating with the back-end is partitioned into four categories: user operations, list operations, item operations and shop operations. The operations used by the Web interface are summarised as fol-lows and listed in detail in Table 1 to Table 4.

User operations

- **`User_AddReminder`** – adds a reminder to the user's list
- **`User_AddShop`** – adds a shop to a user's favourite shops list
- **`User_AddTrustedUser`** – adds a user to a user's trusted user list
- **`User_ChangePassword`** – changes the user's password
- **`User_Create`** – creates a new user
- **`User_DeleteReminder`** – deletes a user's reminder
- **`User_DeleteShop`** – removes a shop from a user's favourite shops list
- **`User_DeleteTrustedUser`** – removes a user from a user's trusted user list
- **`User_Details`** – retrieves a user's details (user name, e-mail address, etc.)
- **`User_Edit`** – modifies a user's details
- **`User_EditReminder`** – modifies a reminder's details
- **`User_GetDeletedItems`** – retrieves a list of items the user has deleted from their shopping lists in the current session
- **`User_GetLists`** – retrieves a list of a user's shopping lists
- **`User_GetNotifications`** – retrieves a user's notifications (news)
- **`User_GetReminders`** – retrieves a list of a user's reminders
- **`User_GetShops`** – retrieves a user's favourite shops list
- **`User_GetTrustedUsers`** – retrieves a user's trusted user list
- **`User_GivePermission`** – grants a user access to a shopping list
- **`User_Login`** – logs a user into the system
- **`User_Logout`** – logs a user out of the system
- **`User_RevokePermission`** – revokes a user's access to a shopping list
- **`User_SearchUsers`** – searches for a particular user of the system

List operations

- **`List_AddExistingItems`** – adds an item that exists in the system to a list
- **`List_AddItems`** – creates and adds items to a shopping list
- **`List_Create`** – creates a new shopping list
- **`List_Delete`** – deletes a shopping list
- **`List_DeleteItems`** – deletes items from a shopping list
- **`List_Edit`** – modifies the details of a shopping list
- **`List_EditItems`** – modifies the details of an item in a shopping list
- **`List_EditQuantity`** – modifies the quantity of an item in a shopping list
- **`List_GetItems`** – retrieves a list of the items in a shopping list
- **`List_GetUsers`** – retrieves a list of users who have access to a shopping list

Item operations

- **`Item_CheckoutItems`** – marks items as bought
- **`Item_GetCategories`** – retrieves a list of possible item categories

Shop operations

- **`Shop_AddLayout`** – adds a shop layout
- **`Shop_GetAll`** – retrieves a list of shops
- **`Shop_GetLayout`** – retrieves a shop's layout

XML Listing 1

```
<Items><I id="33" name="Butter" quantity="1" category="Butter"
shop="Checkers, Rondebosch, Riverside Centre" shopID="1" private="0"
uncertain="0" user="UserX" userID="2" date="11-10-2007">Ran out
today.</I></Items>
```

XML Listing 2

```
<Items><I id="33" n="Butter" q="1" c="Butter" sh="Checkers,
Rondebosch, Riverside Centre" sid="1" p="0" u="0" usr="UserX" uid="2"
dt="11-10-2007">Ran out today.</I></Items>
```

Figure 18 : Two possible XML structures for a shopping item

Table 1 : The user functions of the API that are used in the Web interface

User Operations	
User_AddReminder(UserId, ReminderName, ReminderDescription, ReminderDate, ReminderPeriod, ReminderItem)	**User_AddShop(UserId, ShopId)**
Function: adds a reminder to the user's list	Function: adds a shop to a user's favourite shops list
Request Type: POST	Request Type: POST
Page Used On: Reminders	Page Used On: Preferences
Example Response: <id>129</id>	Example Response: <id>1</id>
User_AddTrustedUser(UserId, SecondUserId)	**User_ChangePassword(UserName, UserPassword, NewUserPassword)**
Function: adds a user to a user's trusted user list	Function: changes the user's password
Request Type: POST	Request Type: POST
Page Used On: Preferences	Page Used On: Preferences
Example Response: <id>1025</id>	Example Response: 1 *Note: This response should be in XML, bust is not due to oversight. This needs to be corrected in the future.*
User_Create(UserName, UserPassword, UserFirstName, UserSurname, UserEmail, UserCellphone)	**User_DeleteReminder(UserId, ReminderId)**
Function: creates a new user	Function: deletes a user's reminder
Request Type: POST	Request Type: POST
Page Used On: Register	Page Used On: Reminders
Example Response: <id>1025</id>	Example Response: <id>128</id>
User_DeleteShop(UserId, ShopId)	**User_DeleteTrustedUser(UserId, SecondUserId)**
Function: removes a shop from a user's favourite shops list	Function: removes a user from a user's trusted user list
Request Type: POST	Request Type: POST
Page Used On: Preferences	Page Used On: Preferences
Example Response: <id>1</id>	Example Response: <TrustedUsers> <u id="1000" n="bobjones" f="Robert" s="Jones" e="bob@jones.com" c="5554202"></u> ... </TrustedUsers>
User_Details(UserId)	**User_Edit(UserId, UserName, UserFirstName, UserSurname, UserEmail, UserCellphone)**
Function: retrieves a user's details (user name, e-mail address, etc.)	Function: modifies a user's details
Request Type: GET	Request Type: POST
Page Used On: Preferences	Page Used On: Preferences
Example Response: <u id="3" n="HunterU" f="Graham" s="Hunter" e="hunter@test.com" c="55542024"/>	Example Response: 1 *Note: This response should be in XML, bust is not due to oversight. This needs to be corrected in the future.*
User_EditReminder(UserId, ReminderId,ReminderName, ReminderDescription, ReminderDate, ReminderPeriod, ReminderItem)	**User_GetDeletedItems(UserId)**
Function: modifies a reminder's details	Function: retrieves a list of items the user has deleted from their shopping lists in the current session
Request Type: POST	Request Type: GET
Page Used On: Reminders	Page Used On: List Edit
Example Response: <id>128</id>	Example Response: <Items><I id="-1" n="Meat" q="0" c="" sh="" sid="-1" p="0" u="0" usr="null" uid="-1" dt="15-10-2007">_</I></Items> *Note: The "_" indicates a blank note. There was a problem parsing a parameter string with a blank note parameter on the back-end. This is meant to serve as a temporary fix until the issue can be resolved in the future.*
User_GetLists(UserId)	**User_GetNotifications(UserId)**
Function: retrieves a list of a user's shopping lists	Function: retrieves a user's notifications (news)
Request Type: GET	Request Type: GET
Page Used On: News, Lists	Page Used On: News
Example Response: <Lists> <list id="3" uid="3" n="newlist" access="111" s="Checkers, Rondebosch, Riverside Centre" sid="1"/> <list id="4" uid="3" n="Favourites" access="111" s="Checkers, Rondebosch, Riverside Centre" sid="1"/> <list id="2004" uid="3" n="PreviousItems" access="111" s="" sid="-1"/> ... </Lists>	Example Response: <Notifications> <Updates/> <Reminders> <r id="128" n="Electricity" d="Pay bill" date="16-10-2007" p="3" i="1"/> </Reminders> </Notifications>

User_GetReminders(UserId)	**User_GetShops(UserId)**
Function: retrieves a list of a user's reminders	Function: retrieves a user's favourite shops list
Request Type: GET	Request Type: GET
Page Used On: Reminders	Page Used On: Lists, List Edit, Preferences
Example Response: <Reminders> <r id="128" n="Electricity" d="Pay bill" date="16-10-2007" p="3" i="1"/> </Reminders>	Example Response: <Shops> <Shop id="1" n="Checkers" l="Riverside Centre" c="Cape Town" s="Rondebosch"/> ... </Shops>
User_GetTrustedUsers(UserId)	**User_GivePermission(UserId, SecondUserId ListId, AccessRights)**
Function: retrieves a user's trusted user list	Function: grants a user access to a shopping list
Request Type: GET	Request Type: POST
Page Used On: Lists, Preferences	Page Used On: Lists
Example Response: <TrustedUsers> <u id="505" n="mark" f="Mark" s="Jacobs" e="now@then.com" c="4534576"/> ... </TrustedUsers>	Example Response: <id>1000</id>
User_Login(UserName,UserPassword)	**User_Logout(UserId)**
Function: logs a user into the system	Function: logs a user out of the system
Request Type: POST	Request Type: GET
Page Used On: Index (log in page)	Page Used On: All except Index
Example Response: <u id="1006" n="harry" f="harry" s="harry" e="harry" c="2345324"></u>	Example Response: <id>3</id>
User_RevokePermission(UserId, SecondUserId, ListId)	**User_SearchUsers(SearchString, UserId)**
Function: revokes a user's access to a shopping list	Function: searches for a particular user of the system
Request Type: POST	Request Type: GET
Page Used On: Lists	Page Used On: Register, Preferences
Example Response: <id>1000</id>	Example Response: <Users> <u id="1000" n="bobjones" f="Robert" s="Jones" e="bob@jones.com" c="5554202"></u> ... </Users>

Table 2 : The list functions of the API that are used in the Web interface

List Operations	
List_AddExistingItems(ListId, UserId, ItemIdList)	**List_AddItems(ListId, UserId, ItemNameList, ItemCategoryList, ItemNote, ItemUncertain, ItemPrivate, ShopId, ItemQuantity)**
Function: adds an item that exists in the system to a list	Function: creates and adds items to a shopping list
Request Type: POST	Request Type: POST
Page Used On: List Edit	Page Used On: List Edit
Example Response: <id>1664</id>	Example Response: <id>1663</id>
List_Create(ShopId, ListName, UserId)	**List_Delete(ListId, UserId)**
Function: creates a new shopping list	Function: deletes a shopping list
Request Type: POST	Request Type: POST
Page Used On: Lists	Page Used On: Lists
Example Response: <id>2085</id>	Example Response: <id>2086</id>
List_DeleteItems(ListId, UserId, ItemIdList)	**List_Edit(ListName, ListId, UserId, ShopId)**
Function: deletes items from a shopping list	Function: modifies the details of a shopping list
Request Type: POST	Request Type: POST
Page Used On: List Edit	Page Used On: Lists
Example Response: <id>1449</id>	Example Response: <id>2085</id>
List_EditItems(ListId, UserId, ItemIdList, ItemNameList, ItemCategoryList, ItemNote, ItemUncertain, ItemPrivate, ShopId, ItemQuantity)	**List_EditQuantity(ListId, UserId, ItemIdList, ItemQuantity)**
Function: modifies the details of an item in a shopping list	Function: modifies the quantity of an item in a shopping list
Request Type: POST	Request Type: POST
Page Used On: List Edit	Page Used On: List Edit
Example Response: <id>1663</id>	Example Response: <id>1474</id>

List_GetItems(ListId, UserId, OrderBy)	**List_GetUsers(ListId, UserId)**
Function: retrieves a list of the items in a shopping list	Function: retrieves a list of users who have access to a shopping list
Request Type: GET	Request Type: GET
Page Used On: List Edit	Page Used On: Lists
Example Response: <Items> <I id="33" n="Butter" q="1" c="Butter" sh="Checkers, Rondebosch, Riverside Centre" sid="1" p="0" u="0" usr="UserX" uid="2" dt="11-10-2007"> Ran out today.</I> ... </Items>	Example Response: <Users> <u id="3" n="HunterU" f="Graham" s="Hunter" e="hunter@test.com" c="55542024" access="111"/> ... </Users>

Table 3 : The item functions of the API that are used in the Web interface

Item Operations	
Item_CheckoutItems(ItemId, ListId, UserId)	**Item_GetCategories()**
Function: marks items as bought	Function: retrieves a list of possible item categories
Request Type: POST	Request Type: GET
Page Used On: List Edit	Page Used On: List Edit, Shop Layouts
Example Response: <id>1458</id>	Example Response: <Categories> <c id="4">Reminder</c> <c id="5">Baby Care</c> ... </Categories>

Table 4 : The shop functions of the API that are used in the Web interface

Shop Operations	
Shop_AddLayout(ShopId,UserId,Layout)	**Shop_GetAll()**
Function: adds a shop layout	Function: retrieves a list of shops
Request Type: POST	Request Type: GET
Page Used On: Shop Layouts	Page Used On: Shop Layouts, Preferences
Example Response: <id>8</id>	Example Response: <Shops> <Shop id="1" n="Checkers" l="Riverside Centre" c="Cape Town" s="Rondebosch"/> ... </Shops>
Shop_GetLayout(ShopId, UserId)	
Function: retrieves a shop's layout	
Request Type: GET	
Page Used On: Shop Layouts	
Example Response: <Layouts> <Layout sid="1" id="1" uid="3"> <aisle id="1"> <s>10</s><s>63</s><s>72</s> </aisle> <aisle id="2"> ... </Layout> </Layouts>	
Data format: a layout comprises a number of aisles. Each aisle has a number of sections (<s></s>), each of which represents a single item category and contains that category's system ID.	

CHAPTER 4

EVALUATION

The Web interface was evaluated through user testing and heuristic evaluation. This chapter describes the design of each evaluation, lists the results gathered and then discusses these results.

4.1. DESIGN OF EVALUATION

4.1.1. User Testing

The user testing was a direct observation evaluation [19], which yielded qualitative results.

The users who were involved in the evaluation fell into one of three classes:

1. The interface "experts" who had been involved in the design phases earlier in the project.
2. People who are knowledgeable of computers and technology.
3. People who are considered computer novices.

Eight users evaluated the system: three from the first class, two from the second and two from the third. The users were asked (verbally, by the author and according to a questionnaire) to perform a series of tasks and the author observed them, noting any problems they had while performing the tasks, as well as any faults that they discovered in the system. As they performed the tasks, they were also asked to interpret the system's feedback – for instance, what certain icons mean.

Before the actual evaluation began a pilot study was performed using two people. The aim to was to check that the users would understand what was being asked of them in the evaluation. The pilot study led to several minor changes to the phrasing of questions and ordering of tasks, and one major change, which involved the My Shop Layouts page.

The question involving this page asked the evaluator to lay out a shop using a given floor plan. However, the first user didn't understand the section's description (on the Web site), which explained what the user is meant to do with it. After having it explained to them, they understood and suggested that the description be re-written. So it was, before the second user performed the evaluation. However, this user had the same problem. They suggested that instead of a textual description, the Web site rather have an example floor plan and it's corresponding Web layout. However, there was not enough time to implement this, so it was decided that the evaluation would be changed. Instead of asking each tester to lay out a shop, the feature was instead discussed with them in order to learn how they visualise a shop's layout and find out how they would like the feature to function.

The final version of the evaluation questionnaire is attached as Appendix 8.1.

4.1.2. Heuristic Evaluation

Heuristic evaluation is a technique created as a way of structuring expert evaluation of an interface [19]. The idea is that the experience of interface designers is condensed into a number of design heuristics against which the interface is evaluated by a usability expert. It was created by Nielsen, who proposed 10 general heuristics (Figure 19) [28]. During the evaluation, the expert examines the system and rates each heuristic using Nielsen's severity ranking scale (Figure 20) [27].

Three users performed individual heuristic evaluations. All of them are Computer Science students; two are PhD students and third is a second-year Masters student. All three are working on projects that revolve around usability. They used Nielsen's heuristics and ranking scale in their evaluations. The heuristic evaluation sheet template is included is included as Appendix 8.2.

Nielsen's 10 Heuristics

1. Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Figure 19 : Nielsen's 10 heuristics

Nielsen's Severity Ranking Scale

Rating	*Description*
0	I don't agree that this is a usability problem at all.
1	Cosmetic problem only. Need not be fixed unless extra time is available on project.
2	Minor usability problem. Fixing this should be given low priority.
3	Major usability problem. Important to fix, so should be given high priority.
4	Usability catastrophes. Imperative to fix this before product can be released.

Figure 20 : Nielsen's severity ranking scale

4.2. RESULTS

4.2.1. User Testing

As is to be expected, some users had more trouble using the interface than others. However, there were several problems that repeatedly cropped up:

1. The interface does not convey that it is communicating with the server and waiting for a response. This, and the fact that there was often a delay during this communication, meant that many of these users believed that they had not pressed a button or done something properly, and clicked the button again or began to re-do the operation.

2. Most of the users spent time searching the screen to find functions, particularly in the page where one edits the shopping list.
3. Most users did not understand what the right-hand pane on the shopping list editor page is for (Figure 21).
4. There is no cursor shown when one clicks on the "note" text area of the "edit item" dialog (Figure 22). This resulted in every user who used the screen clicking the area multiple times to try to select it, when in fact the area was selected on their first click.
5. The "Add a new reminder" form has a field with the label "Name:" (Figure 23). Several users mistook this to mean their name, rather than the name of the reminder.
6. Some users tried to edit the date field of the "Add a new reminder" form (Figure 24), rather than use the calendar (accessed through the button beside the field) to select a date (Figure 25).
7. The shop layouts screen was a big problem. The description text generally was not understood.

The user testing also uncovered a number of bugs, though none caused the system to be unusable. Many are simple inconsistencies and can be easily corrected. An example is: when a new list is added, its "edit" and "add trusted users" buttons do not have tooltips (they should); however, if the page is reloaded, they do have these tooltips.

4.2.2. Heuristic Evaluation

Figure 26 lists the ratings of the three heuristic evaluations and Figure 27 lists some of the evaluators' comments. The evaluators' full evaluation feedback sheets are attached as Appendix 8.3. (The evaluators filled the sheets out on computer, so each sheet in the appendix is a verbatim copy of their feedback.)

4.3. DISCUSSION OF RESULTS

Both sets of evaluators were happy with the interface on the whole. Most of them liked its visual design and thought it generally functioned well.

The lack of feedback on server communication was an unfortunate oversight. It is a clear violation of Nielsen's first heuristic, which is a usability fundamental: keep the user informed of what the system is doing. Rectifying this is a vital piece of future work.

The fact that many users searched the screen to find functions is a difficult problem to solve. An obvious solution is to provide more immediate guiding help on the screen. However, this will quickly clutter the interface, especially given that there is limited space on the screen. But it might be viable to provide lots of on-screen help by default, but allow the user to hide this help by selecting an option in their preferences (or one could give them the option of "minimising", thus hiding, the help on a per-screen basis). The aim of this would be to allow flexibility so that novice users can easily learn the system and advanced users can use it without being bogged down. Currently the system is more geared toward the latter (see Evaluator 1's comments on heuristic 7, Figure 27 above).

The lack of understanding with the right-hand pane on the shopping list editor page can definitely be solved by adding an appropriate description to the page.

The lack of a cursor when editing an item's note in the "edit item" dialog is a known Firefox bug [5], which has no definitive work-around solution. The bug has been fixed, but not in the current version of Firefox (version 2); the solution will only appear in Firefox version 3. The problem does not occur in Internet Explorer. (Note: all of those involved in the user testing used Firefox, hence they all encountered this problem.)

The "Add a new reminder" form "name" field problem is easy to solve – one simply

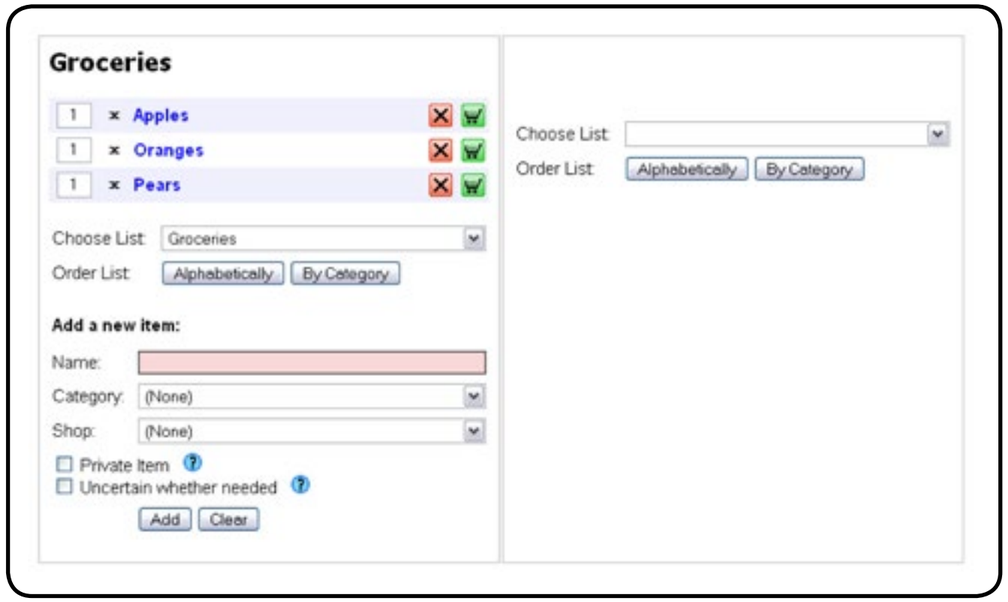


Figure 21 : The List Edit page, showing the right-hand pane

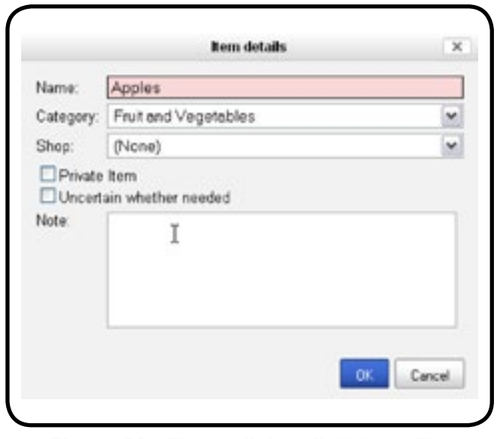


Figure 22 : The "edit item" dialog. The "Note" area has been clicked, but there is no cursor

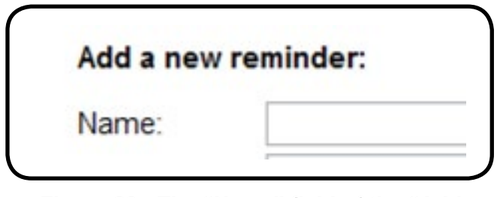


Figure 23 : The "Name" field of the "Add new reminder" form

Figure 24 : The non-editable date field of the "Add a new reminder" form, with the calendar access button next to it

Figure 25 : The date-picking calendar of the "Add a new reminder" form

changes the label to "Reminder Name". When designing the form, it was believed that a user would automatically associate the field with the reminder they were adding. Some users did. Those who did not were not all one type of user: users from each class made the mistake. This shows how easy it is to assume that what is obvious to one person is obvious to everyone.

The reminder date selection problem can also be corrected by adding on-screen help (an instruction to use the calendar). However, it was not as widely misinterpreted as the reminder name field label. The date field is a non-editable one and is there primarily to provide feedback from the calendar dialog. Most of the experienced users recognised this and clicked the calendar button; it was the novice users, in particular, who did not realise this.

One heuristic evaluator commented that the category list is perhaps too long and needs to be broken up. However, breaking it up is difficult – what would one break it up into? But the comment does have merit: the list currently consists of 73 items and is by no means exhaustive (the categories are those of the Pinelands Pick 'n Pay, which is grocery-orientated; the larger "all-in-one" stores will have far more categories).

The largest problem is clearly the shop layouts page (Figure 28). Nearly everyone in both sets of evaluators struggled with it. The layout concept is difficult to convey through a textual description. Undoubtedly a visual example should be included. However, the layout system also needs to be worked on and improved: it is currently inflexible (e.g., there is no way to delete an aisle) and not very user-friendly (the user feedback is not particularly useful, as the heuristic evaluators indicated).

	Evaluator		
Heuristic	**1**	**2**	**3**
1	0	3	1
2	1, 3	1	0
3	0	2	1
4	2, 1	1	0
5	1	1	2
6	2	0	1
7	1, 0	1	2
8	0	0	0
9	0	3	3
10	5	1	2

Figure 26 : The usability experts' ratings for each heuristic

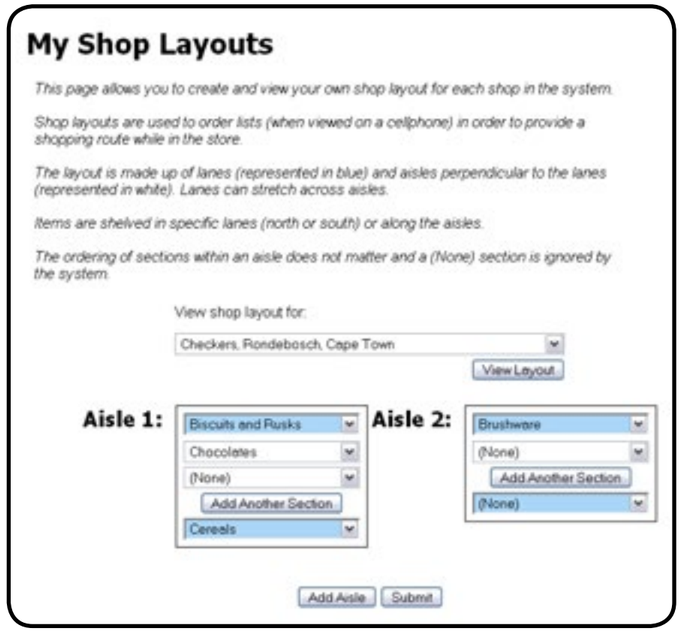
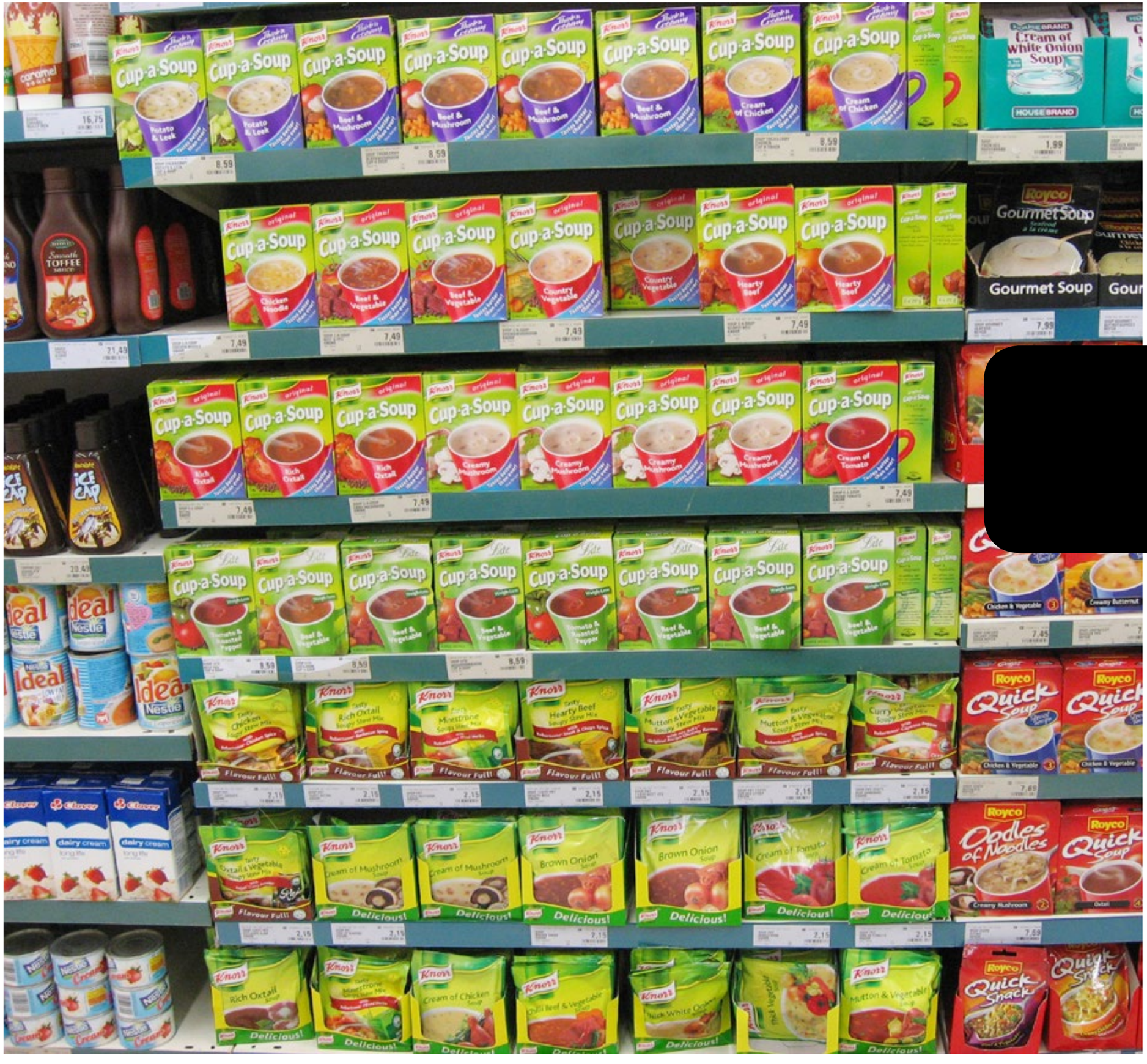


Figure 28 : The problematic My Shop Layouts page

Evaluator 1	Evaluator 2
2: The aisle list model is a good idea for simplicity sake but you will need a graphical layout, there is no way around this	1: Login took a while and there was no indication that my login was being processed; all the pages should have descriptions at the top, somewhat like the MyShopLayouts page.
5: Need a way to remove an aisle, also the system crashed when I added 6 aisles other wise the error prevention seems good throughout the site	2: I found some of the description on MyShopLayouts a little unclear (about aisles and lanes). When I tried to add an aisle, this confusion was continued. E.g. the drop down lists above and below the Add Another Section button are confusing. What does the bottom one refer to? Are they north and south – but then when you add a section it only goes on top?
2: Long lists of item categories are not ideal, is there someway to break the lists into more manageable segments	3: There is no cancel or undo button for changing something done by mistake, e.g. where one creates a shop layout and aisle sections
7: Where do I start the process? You need to control the initial experience of the workflow and eliminate any guesswork for new users	9: There is no error message if I press ViewLayout on the MyShopLayouts page without selecting a shop. Nothing happens. When I tried to create a shop layout (by setting up an aisle and pressing Submit), I was unable to and got an error message which was very system specific and technical (Error Null, xml parsing error…)
7: System use is very flexible which is a good feature as the experienced users are able to streamline their use of the system	
10: Main Screen – as this is a Web page there are many ways a user could have found it – there needs to be some quick overview documentation – to set the scene	
Evaluator 3	
5: The creation of shop layout is prone to error since it is not clear that aisle must be created and added one at a time	9: The error messages that accompany errors in the layout creation are not helpful
7: This is not a problem in most of the system but the layout creation is very restrictive visually	10: The layout creation description is unclear

Figure 27 : A selection of the heuristic evaluators' comments
(Each number denotes the heuristic the comment is about)



CHAPTER 5

CONCLUSIONS

The key aim of the Cellphone Shopper project is to make grocery shopping easier by using technology to aid the process. To accomplish this, the project was divided into three parts: a back-end, a Web user interface and a cellphone interface.

I was assigned to handle the Web inter-face. The project proposal stated that I had two sets of tasks:

- To design and implement the Web-based end-user interface
- To perform user evaluations of the in-terface

I have done both over the span of the project.

The proposal also listed the following as the key success factors for the Web interface:

- The features specified for the system are fully implemented and function prop-erly.
- Users consider the interface to have good aesthetics and be easy to use.
- The interface has good performance and can bear an acceptable workload.

The majority of the key features pro-posed for the system as whole have been implement-ed. The following are missing:

- Pre-population of the user's shopping list based on their list history. This was not implemented because the initial user interviews, held to determine what users would like in the system, revealed that most users do not desire to have it.
- The ability to view statistics about the list history. The project team decided that this feature was not a high priority one. Because of this and the fact that there was limited time to work on the project, we never got around to imple-menting it. However, I feel it is some-thing that can easily be added to the sys-tem later.

In addition to the proposed core fea-tures, several suggested by users and evalua-tors were implemented. These include re-minders, favourite shops and trusted users. The project team felt that these were useful and interesting features worth including in the system.

The Web interface is by no means per-fect, as the evaluations showed. Nor is it entirely bug free. However, its implementa-tion is largely successful. Evaluations showed that most of the interface is easy to use and many of the evaluators commented on its good aesthetics.

Finally, though not thoroughly tested, the interface appears to perform well. Server delays are the only performance problem, and this could most likely be rectified by moving the back-end and interface on to a server out-side of the UCT network.

CHAPTER 6

FUTURE WORK

This chapter lists improvements that could be made to the system, particularly the Web interface, as well as possible features that could be added to it. Some of these were suggested in the design interviews and evaluations; the sourced is noted where possible.

News

- Add an option to set whether the system clears the news automatically or the user does so manually. [Design interview 2]
- Warn the user when they log into the system if someone is doing shopping using one of the lists they have access to. [Design interview 2]

Reminders

- Have an option to renew reminder automatically or manually. [Design Interview 3]
- Reminders should perhaps be kept until the user deletes them. Just because the reminder has passed, it does not mean that the user saw it; nor does it mean that they do not want to see it now. [Evaluation pilot study]

Layout

- The layout functionality needs to be more flexible. It should allow deleting and moving aisles, and perhaps factor in the shops' entrance(s) and exit(s).
- There should be default layouts if possible. Either the system administrators could create these themselves or they could choose from what users have already created.

List

- List synchronisation needs to be handled properly. If (for example) two users are concurrently editing a list, the system does not update each user's list when the other user makes a change. The only way for the users to see these changes is for them to manually refresh the list.
- Allow deleted items to be added back. Currently, when an item is deleted it becomes a "non-item" – it becomes a temporary object in the system that persists only for a the user's current session. Because of this, it cannot be added back in the current system.
- Introduce a priority attribute for items. This could be binary (either the item is a priority or it isn't) or multi-valued (i.e., there is a rating).

Preferences

- Allow more flexible shop management – allow the user to specify a shop's name, city, suburb, location, rather than simply list a number of shops. This would allow users to add their own shops, something which is particular important given that new shops are always being built.

Help

- Build on the current help by including more descriptions. In particular, have a description for each page.
- Possibly provide lots of on-screen help by default, but allow the user to hide this help by selecting an option in their preferences.

Security

- The current back-end allows direct querying – one just has to know the correct parameters and values. It should be secured so that only logged in users may access information, and only information that applies to them.
- It may be necessary to improve the login system's security. Currently, only the user's password is encrypted – login requests and responses are not, nor are the user names within them.

Testing

- The system has not been tested on a wide scale using many users and over a long period of time. There are bound to be "cross-interaction" errors that have not been discovered. So a longer and bigger evaluation could be conducted.

Optimisation

- Each operation on the Web interface that requires a corresponding server operation leads to individual communication with the server. It should be possible to batch server requests and thus lower server and Web traffic.

Additional interfaces

- The ability to use a camera to photograph barcodes, which the system can decode. Items could then be identified by their barcodes, which means a whole new range of possible functions. For example, items can be marked off as "bought" simply photographing their barcodes as they are put into the trolley.
- Alternatively, camera-based recognition of the product itself.
- A WAP-like cellphone interface could be developed to broaden the range of cell-phones that can use the system (because not all phones can run J2ME applica-tions).
- Applications could be developed for the Microsoft Windows Mobile and Sym-bian OS platforms.
- The cellphone application could be extended to include the ability to edit the store layout. Editing the shop layout while in the store would be far more convenient than having to draw it out and construct the layout later on a computer.

Other

- Better user feedback on the Web interface during server communication. Loading notices and animations should be implemented to provide better user feedback.
- History statistics and other functions could be implemented once the database has been built up.
- The system could be extended to support prices. This, however, would require the shopping chains, or even individual stores, to buy into the system, as they control and manage the prices. Introducing prices means that one could begin to analyse sales data. However, this alone could dissuade shopping chains from providing their prices – immediate sales statistics are probably regarded as private. (Many of these companies are public, so their financial year sales figures are publicly available. However, the knowledge of how much they sold last week, for instance, is something would probably want to keep from anyone, especially competitors).
- Community features would be useful – the ability to inform others about vari-ous things, like a shop being out of stock of an item or item specials and discounts.
- "State" management functionality, particularly the management of the lifetimes of products (to warn of expiration), could be introduced.

CHAPTER 7

REFERENCES

[1] AjaxTrans, http://www.ajaxtrans.com/.

[2] Amazon.com, http://www.amazon.com/.

[3] Aziz, A., and Kollhof, J. 2006. JSON-RPC 1.1 Specification. Working draft. August. Web site: http://json-rpc.org.

[4] Bellamy, R., Brezin, J., Kellogg, W.A., Richards, J. and Swart, C. Designing an E-Grocery Application for a Palm Com-puter: Usability and Interface Issues. IEEE Personal Communications, 8, 4, 60-64, August 2000.

[5] Bug 167801 – Cursor (caret) sometimes fails to appear in input text fields (shown/painted in wrong widget). Re-trieved 14 October 2007 from Bugzilla at Mozilla: https://bugzilla.mozilla.org/show_bug.cgi?id=167801.

[6] Critical Issues for Web site Develop-ment. Retrieved 8 July 2007 from Net Access: http://www.netxs.com.pk/web/critical_issue_for_web_develop.html.

[7] del.icio.us, http://del.icio.us/.

[8] Facebook, http://www.facebook.com/.

[9] Folmer, E. 2005. Software Architecture analysis of Usability. PhD thesis, University of Groningen, Mathematics and Computer Science.

[10] Frameworks. Retrieved 8 July 2007 from Ajax Patterns: http://www.ajaxpatterns.org/Frameworks.

[11] Garrett, J. J. 2005. AJAX: A new approach to web applications. Retrieved 8 July 2007 from Adaptive Path: http://www.adaptivepath.com/publications/essays/archives/000385.php.

[12] Google Calendar, http://calendar.google.com/.

[13] [13] Google Mail, http://www.gmail.com/.

[14] [14] Google Maps, http://maps.google.com/.

[15] Google Suggest, http://www.google.com/webhp?complete=1&hl=en.

[16] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., and Lafon, Y. 2007. SOAP Version 1.2 Part 1: Messaging Framework (Sec-ond Edition). W3C Recommendation. April. World Wide Web Consortium (W3C). Web site: www.w3.org.

[17] Hixie, I. 2005. Call an apple an apple. Retrieved 10 July 2007 from Hixie's Natural Log: http://ln.hixie.ch/?start=1111339822.

[18] Introducing JSON. Retrieved 10 July 2007: http://json.org/.

[19] Jones, M. and Marsden. 2006. G. Mo-bile Design Interaction, John Wiley & Sons, Ltd..

[20] JSON: The Fat-Free Alternative to XML. Retrieved 10 July 2007: http://www.json.org/xml.html.

[21] Lerner, R. M. 2001. At the Forge: Introducing SOAP. Linux Journal. 2001, 83es (Mar. 2001), 11.

[22] LiveMarks, http://sandbox.sourcelabs.com/livemarks/.

[23] Live Search, http://www.live.com.

[24] Mesbah, A. and van Deursen, A. 2007. An Architectural Style for Ajax. In Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture (January 06 - 09, 2007). WICSA. IEEE Com-puter Society, Washington, DC, 9.

[25] Morin, R. SOAP, REST and XML-RPC. 2006. Retrieved 10 July 2007 from The RSS Blog: http://www.kbcafe.com/rss/?guid=20060704042846.

[26] Murray, J., Schell, D., and Willis, C. 1997. User centered design in action: developing an intelligent agent application. In Proceedings of the 15th Annual international Conference on Computer Documentation (Salt Lake City, Utah, United States, October 19 - 22, 1997). SIGDOC '97. ACM Press, New York, NY, 181-188.

[27] Nielsen, J. 2005. Severity Ratings for Usability Problems. Retrieved 14 Oc-tober 2007 from useit.com: http://www.useit.com/papers/heuristic/severityrating.html.

[28] Nielsen, J. 2005. Ten Usability Heuristics. Retrieved 14 October 2007 from useit.com: http://www.useit.com/papers/heuristic/heuristic_list.html.

[29] Panic Goods, http://www.panic.com/goods/.

[30] [30] Ruby, S. 2003. XML-RPC, SOAP, and/or REST. Retrieved 10 July 2007 from Intertwingly: http://intertwingly.net/blog/1507.html.

[31] Stamey, J. and Richardson, T. 2006. Middleware development with AJAX. Journal of Computing Sciences in Colleges. 22, 2 (Dec. 2006), 281-287.

[32] Teo, H., Oh, L., Liu, C., and Wei, K. 2003. An empirical study of the effects of interactivity on web user attitude. International Journal of Human-Computer Studies. 58, 3 (Mar. 2003), 281-305.

[33] Turner, A., and Wang, C. 2007. AJAX: Selecting the Framework that Fits. Retrieved 9 July 2007 from Dr. Dobb's Portal: http://www.ddj.com/dept/webservices/199203087?pgno=1.

[34] Wayner, P. 2006. Surveying open-source AJAX toolkits. Retrieved 9 July 2007 from InfoWorld: http://www.infoworld.com/article/06/07/31/31FEajax_1.html.

[35] Woolworths, http://www.woolworths.co.za.

[36] Wu, H. and Natchetoi, Y. 2007. Mobile shopping assistant: integration of mobile applications and web services. In Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM Press, New York, NY, 1259-1260.

[37] XML-RPC Home Page. 2003. Retrieved 10 July 2007: http://www.xmlrpc.com/.

[38] XmlRpcDiscussion. 2006. Retrieved 10 July 2007 from Intertwingly: http://intertwingly.net/wiki/pie/XmlRpcDiscussion.

CHAPTER 8

APPENDICES

8.1. EVALUATION QUESTIONS

CELLPHONE SHOPPER

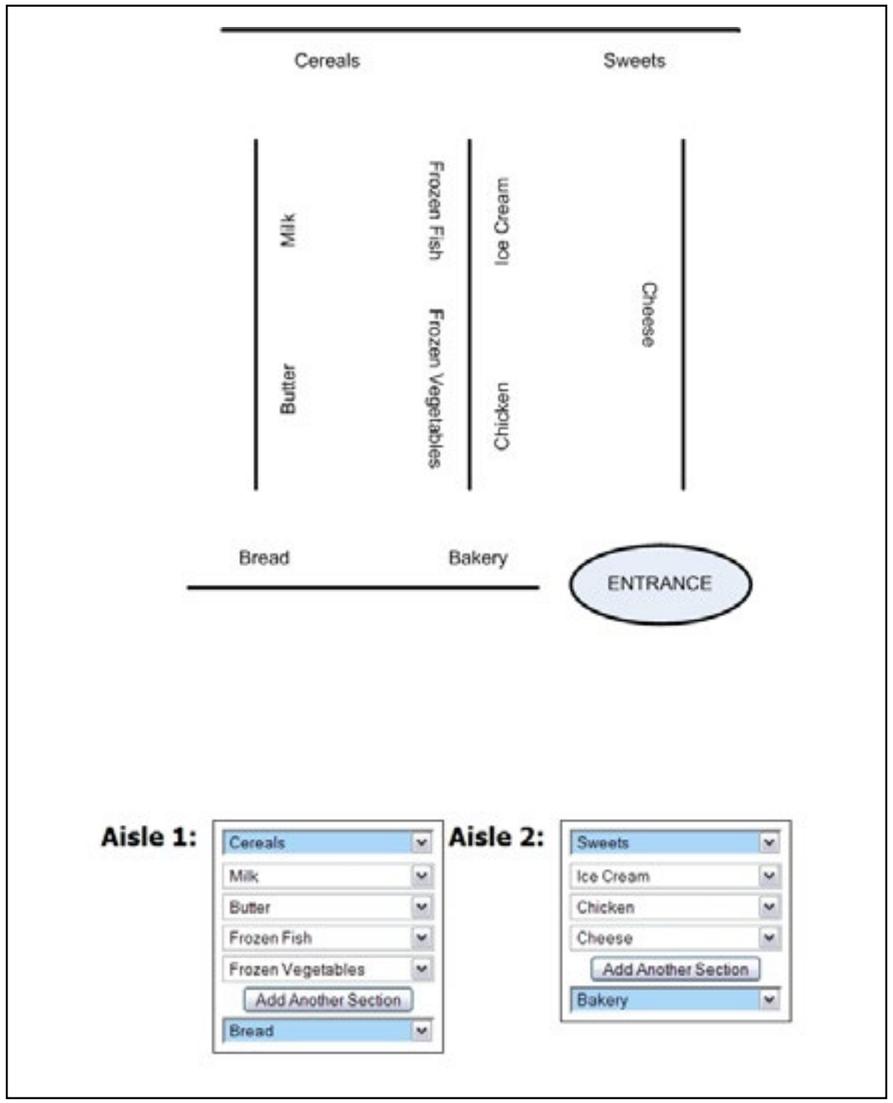
Evaluation Tasks and Questions

Person:

Evaluation Number:

Pre-evaluation: explain the system.

1. Login as “eval##” using the password “eval##”.
2. What news is there?
3. Explain the My Preferences section.
4. Add a reminder to remind yourself to pay the electricity bill on 25 October.
5. Add a new list – you can give it any name.
6. Grant Robert Jones access to it.
7. Go into the list. Add an item to it. Now edit the item and attach a note to it.
8. How do you mark the item as bought? How do you check to see who added the item?
9. Add “Apples” from your Favourites list to the list you created.
10. Select the list “List X” on the left-hand side. What does the yellow R square mean? Who added the reminder?
11. Discuss the My Shop Layout screen using the store layout example.



8.2. HEURISTIC EVALUATION SHEET

Heuristic Evaluation

Heuristics

1. Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Evaluation

Name:

Nielsen's severity ranking scale (SRS)

Rating	*Description*
0	I don't agree that this is a usability problem at all
1	Cosmetic problem only. Need not be fixed unless extra time is available on project
2	Minor usability problem. Fixing this should be given low priority
3	Major usability problem. Important to fix, so should be given high priority
4	Usability catastrophes. Imperative to fix this before product can be released

Heuristic	Rating / Notes
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

8.3. FULL HEURISTIC EVALUATION FEEDBACK

Evaluation 1

Heuristic	Rating / Notes
1	0: Feedback of system status is good, ie. When I created a list it was reflected that it was successful.
2	1: Most of the concepts will be familiar to the user but they need to be confident about how and why to use the features. If the Web site is the main configuration portal for the system then much more info needs to be provided before it can be deployed. 3: The aisle list model is a good idea for simplicity sake but you will need a graphical layout, there is no way around this. For your honours project the drop list model will do but in the future work section of your write up you should mention the use of graphics.
3	0 : The user is able to perform the functions in any order which is good but you will need to provide some guidance initially. Later on users can use the system in any order once they are familiar with it.
4	2 : My Lists appears twice, once under the My Lists Tab and once under My News – Are they the same thing? 1 : I noticed some tooltips floating over some menu options, good idea, but you need to use them consistently throughout your Web page, the three icons shown next to a shopping list need descriptive tooltips.
5	1 : Need a way to remove an aisle, also the system crashed when I added 6 aisles other wise the error prevention seems good throughout the site.
6	2 : Long lists of item categories are not ideal, is there someway to break the lists into more manageable segments.
7	1: Where do I start the process? You need to control the initial experience of the workflow and eliminate any guesswork for new users. 0 : System use is very flexible which is a good feature as the experienced users are able to streamline their use of the system.
8	0 : Good, only the core features are exposed, no clutter.
9	0 : I liked the idea of moving items between my lists and previous items and the fact that you laid them out next to each other – if a user makes a mistake they can pull the item back into their list.
10	5 : Main Screen – as this is a Web page there are many ways a user could have found it – there needs to be some quick overview documentation – to set the scene.

If you have any other comments, suggestions or thoughts, please write them here.
Adding an aisle to the shop layout crashed

Evaluation 2

Heuristic	Rating / Notes
1	3 – Login took a while and there was no indication that my login was being processed; all the pages should have descriptions at the top, somewhat like the MyShopLayouts page (especially MyLists, MyReminders, MyNews). The first page I came to (MyNews) was confusing and what news would be put in there and how was not made clear. Occasionally, the system first said I had nothing, e.g. no lists and then changed it to show my lists, which was a little confusing.
2	1 – Mostly this was understandable and fine. I found some of the description on MyShopLayouts a little unclear (about aisles and lanes). When I tried to add an aisle, this confusion was continued. E.g. the drop down lists above and below the Add Another Section button are confusing. What does the bottom one refer to? Are they north and south – but then when you add a section it only goes on top?
3	2 – One can only navigate by the menu – there is no back button and this would make navigation easier. There is no way to close sections that one opens except by reselecting the menu item (e.g. List Access section which appears on MyLists page). There is no cancel or undo button for changing something done by mistake, e.g where one creates a shop layout and aisle sections.
4	1 – Under MyLists Favourites is clickable to open but PreviousItems is not, which is inconsistent.
5	1 – It seems relatively difficult to make a bad error with this system, and you double check when I try to delete a list. The MyShopLayout page is a little error prone because it is somewhat confusing.
6	0 – No problem here
7	1 – This seems reasonable. I like the front page with a list of lists and the news section (if it is for today's issues?). Adding a long shopping list would take a long time, though, as you have to add each item, select a category, etc. Maybe there could be an editable drop down list with categories pre-connected so that you can quickly select items rather than type them in. (I do like the ability to copy items between lists, though, which would speed up the process)
8	0 – Nice minimalist design
9	3 – There is no error message if I press ViewLayout on the MyShopLayouts page without selecting a shop. Nothing happens. When I tried to create a shop layout (by setting up an aisle and pressing Submit), I was unable to and got an error message which was very system specific and technical (Error Null, xml parsing error…) .
10	1 – The tooltips on the icons next to personal lists don't always appear and the icons are not completely easy to understand.

If you have any other comments, suggestions or thoughts, please write them here.
The system in general seems very usable. The interface is clean and attractive. Some buttons didn't work (e.g. Add Shop and Add User) but I assume this is functionality that you are not implementing and the placing seems appropriate. I set a reminder, which the system lost, but this is back-end I presume, not interface. The ShopLayout screen seems the most problematic.

Evaluation 3

Heuristic	Rating / Notes
1	1 There is no feedback indicating that registering was successful. Would be nice if lists under MyNews and MyLists could be expanded (like file-folder hierarchies)
2	0
3	1 Users are mostly free
4	0
5	2 The creation of shop layout is prone to error since it is not clear that aisle must be created and added one as a time
6	1 The info that appears when hovering the mouse over elements doesn't always show up
7	2 This is not a problem in most of the system but the layout creation is very restrictive visually
8	0
9	3 The error messages that accompany errors in the layout creation are not helpful
10	2 The layout creation description is unclear

If you have any other comments, suggestions or thoughts, please write them here.
Overall the system is nice and I found not many problems with it. The lists are easy to create (though they do disappear and reappear at times) and tie together nicely with reminders and news. The most problematic part is the layout creation – it is very inflexible and it is difficult to map one's cognitive map of a shop onto a row-based series of aisles and lanes. Additionally wrt layouts, after seeing a predefined list of shops I was kind of expecting the system to contain a pre-baked layout for the shops – perhaps this makes the system too restrictive though?