Honours Project Report

Report By:
Tshifhiwa Ramuhaheli
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
tramuhah@cs.uct.ac.za

Supervised by:
Dr Hussein Suleman
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
hussein@cs.uct.ac.za

University of Cape Town // Department of Computer Science

2007

Abstract

Managing a shopping list can be a nightmare, especially if the shopping list is shared among a household. In this report we discuss the use of a Cellphone application to simplify the management of these shopping lists. The different methods of creating mobile device applications are investigated and a comparison between the various mechanisms is made. We also look at the different development environments like J2ME, Symbian and BREW. This was done in order to come up with a solution that will make it much easier to manage shopping lists on small screen. The methodology used to design and develop this type of application is elaborated on. The Cellphone application is developed in Java Micro Edition as this environment offers portability across mobile devices. The report elaborates on the whole application development process from the design phase through a series of prototype sessions to the actual implementation of the system. The application was also tested for usability to make sure that it conforms to the proper standards for designing mobile interfaces. In the end a usable interface is proposed for managing shopping lists. Suggestions are also made for future work that may be done to improve the functionality and usability of the system as whole.

Keywords: Cellphone, Shopping Lists, User Interface and Mobile Applications.

Table of Contents

List of Figures

List of Tables

1. Introduction

The use of Cellphones has increased significantly over the past few years. As a result of these developments, the need for Cellphone applications has increased significantly. These applications are very useful in order to accomplish certain tasks at hand since many people usually carry their Cellphones everywhere they go.

1.1 Problem Definition

There are several problems that can arise from managing shopping lists, especially when one list is shared among different users. These problems include:

- The type of medium used to write the shopping list.
- Having a common location where the list is shared in a household.
- Confusion may arise when changes are made to the list, like adding new items or removing items and no one knows who made the changes.
- Misplacing the shopping list or the list getting lost.
- Leaving the shopping list behind while going to do shopping.
- When doing shopping some items might not be described in a way that will distinguish them from other brands.
- Typical shopping lists are written on a media like paper and rearranging the list is difficult. This will have an impact on the amount of time spent while doing shopping, since users may have to revisit certain sections of the shop when they encounter an item on the list that is located in a section that has already been visited.

1.2 Project Outcomes

This project is aimed at solving the shopping list problems mention above by using modern technologies. The whole shopping process is not going to be automated but it will be improved upon and made easier by allowing users to manage their shopping lists efficiently. The solution is based on having the shopping lists located on a central server and having a household sharing these lists. These shopping lists can be manipulated using two user interfaces, which are the Cellphone interface and the Web interface. The Web interface was developed using modern Web-development techniques and technologies, such as AJAX, and the Cellphone interface was developed in the Micro Edition of the Java Platform (J2ME) [5, 13]. The two interfaces will allow the creation and modification of a user's shopping list. The back end Web application handles the related database changes and synchronization between the interfaces. This report will focus mainly on the Cellphone user interface.

2. Background Chapter

In this section we discuss work related to Cellphone Shopper which is a project that involves the use of a Cellphone application to manage shopping lists.

2.1 Shopping Lists

Different people arrange and also make up their shopping lists in different ways. Generally most people have a way of grouping their shopping lists according to certain criteria like type of product or grouping items based on where the product can be purchased from. Ludford, et al. [6] discuss how people create pre-planned information resources like lists. These pre-planned information resources can then be used at a later time and they also refer to the place where the task is performed. The paper also introduces the idea that mobile devices can be used to provide these information resources. This makes it much easier to access and manipulate these information resources since we are more likely to go everywhere with our mobile devices like cell phones and PDAs rather than walk around with a PC.

Pick 'n Pay [5] and Woolworths [6] each have an online shopping system which manages a shopping list when you select your products from their Web interface. These interfaces are very complex and do not offer much functionality. The representation of these lists on the small screen of a Cellphone will prove to be much more challenging. The way these shopping lists are designed works well if you have a pointing device like a mouse and most cell phones do not offer the luxury of a pointing device except in the case of smart phones that have a touch screen. Therefore presentation of lists like these will have to be modified or redesigned since navigation through the list will be much more challenging on a mobile device.

2.2 Similar Systems

There have been a lot of commercial applications developed in J2ME that use GPRS to transfer data over the network. These applications are often optimised to generate as little traffic as possible on the network, thereby making them very cost effective and popular. The following are examples of some mobile applications that use GPRS.

MXit – This is one of South Africa's most popular mobile phone instant messaging applications. This application allows users to chat with other mobile phone users who have the MXit application. By using this application, users can also chat with users from other online networks like Google Talk, Jabber, MSN messenger and ICQ. A user can send text up to 2000 characters at the cost of a fraction of that for an SMS. This application is developed in J2ME and can also send multimedia files such as music, pictures and videos [15].

Google Mail – This is a mobile phone application that connects to any valid Google mail account. This application can retrieve all emails in the account and can be used to compose new emails. It provides various email functions like search, archive and delete. This application basically provides the same functionality as the Web interface but for mobile devices [16].

Go Talk Mobile – This application works in a similar way to MXit or Google Talk as it provides a way of chatting with other users. This is a J2ME application that can be used to connect to Google mail to check for new emails. This application just checks if they are any new emails. Users can chat with their online contacts from Google mail but can not compose and read emails [17].

All these applications were developed for mobile devices using the J2ME development environment. They all use GPRS or 3G for data transfer which minimises the costs as compared to using SMS.

2.3 Development Environment

Cellphone manufactures are now developing environments which make it possible to develop Cellphone applications for a wide range of phone models as compared to developing for just one particular model. This is all done by deploying Cellphone applications on standard operating systems. The two most common operating systems are Symbian and Binary Runtime Environment of Wireless (BREW) [7]. Cross platform applications are possible by using a small version of Java called Java 2 Micro Edition (J2ME) [5, 7].

To successfully compare applications developed in BREW, J2ME and Symbian, we have to also look at the limitations of these environments. Figure 1 provides an overview of the features that are discussed below.

Memory Size: Symbian and BREW applications can use a couple of Megabytes whereas J2ME applications are typically 30 to 64 Kilobytes. The limits are typically 128K or 256K for over the air installation for both Symbian and BREW [7].

Access to Phone Features: In J2ME, MIDP 2.0 provides support for SMS, MMS, calendars, contacts and Bluetooth in the form of JSR-82 but access to SMS can be obtained also from vendor specific APIs [5]. Symbian and BREW applications can access Bluetooth, SMS, MMS, calendars, contacts, and can sometimes dial the phone [7].

Application Speed: J2ME applications are slower than applications developed for Symbian and BREW since Java applications run on the Kilobyte Virtual Machine (KVM) whereas Symbian and BREW applications are compiled to machine code which runs faster [7].

Data Storage: J2ME applications achieve data storage through a record management system (RMS). The data is stored as individual entries and has to be contained in allotted memory. BREW and Symbian have a built-in database system and also support Java database connectivity (JDBC) as an optional package [7].

Multitasking: BREW OS supports cooperative multitasking where an application has to be developed with this in mind. Symbian is a multitasking Operating System and fully supports multitasking [7].

Cross-Platform Portability: J2ME applications can run on Java-enabled phones and on both Symbian and BREW phones. BREW and Symbian applications can only run on phones with the same environment. Symbian applications are further limited to a certain series of Nokia smart phones and have to be rewritten to cater for another series of smart phones.

Although J2ME does not support as many features as Symbian and BREW, it is platform independent and can therefore determine the success of an application. There are more J2ME compatible phones and it therefore provides a wider coverage of handheld devices. As result of this J2ME was chosen as the development environment of the Cellphone interface for the Cellphone Shopper Project. To show how easy it is to develop platform independent applications in J2ME, Davis, et al. [4] demonstrate the possibilities of application tailoring which helps in porting software to a specific mobile device without manually rewriting the code in J2ME.

The following table summaries the comparisons between J2ME, Symbian and BREW.

	J2ME	**Symbian**	**BREW**
Memory Size	30-64 K.	Few Megabytes.	Few Megabytes.
Access to Phone Features	Bluetooth, SMS, MMS, calendars and contacts.	Bluetooth, SMS, MMS, calendars and contacts.	Bluetooth, SMS, MMS, calendars and contacts.
Application Speed	Slower than Symbian and BREW.	Faster than J2ME.	Faster than J2ME.
Data Storage	Record Management System (RMS).	Has a simple built-in Database System.	Has a simple built-in Database System.
Multitasking	-	Fully supports multitasking.	Supports cooperative multitasking.
Cross-Platform Portability	Runs over Java enabled phones, Symbian and BREW phones.	Runs on Symbian Operating System only.	Runs on BREW Operating System only.

Table 1: Comparisons of Development Environments.

2.4 Network Connectivity

J2ME applications can use GPRS for sending data across a network and it is relatively cheaper compared to using normal SMS. Therefore applications using GPRS can benefit by minimizing the amount of network traffic generated by the application. This approach was used in topic map queries from mobile devices in [9]. The speed of data transfer is an important issue and optimizations will have to be made to deal with transfer speeds. This is a common problem and choosing the right protocol will help in minimizing the amount of network traffic between an application and the server.

Bennet, et al. [1] conducted a study where a handheld device was used as a client in order to interact with a message board located on a server. The communication between the J2ME client and the server was through HTTP connection using POST and GET commands. The handheld device was able to connect to the JSP servlet and the response obtained from the server was in a reasonable time. Another study by Wesson, et al. [12] also recommended using J2ME as it is suitable for mobile device applications when bandwidth is a problem or if multiple server requests would be undesirable. The paper also recommends using J2ME if a user wishes to minimize data transfer costs.

Black, et al. [2] had conflicting results in a study of collaborative learning tasks. Handheld devices were used by students to communicate with each other either using voice or textual input. The client application was also developed using J2ME but had slow access to files from the server and this did not bode well for classroom use. Children could become frustrated with such a slow environment. The J2ME documentation caters for this problem but the recommendations made did not solve the problem as there was no change in downloading times.

This study has different requirements from the two studies mentioned above when it comes to the response times achieved when using J2ME applications, which might have been caused by the protocol used and also the amount of data being transferred between the client and server. Choosing the right network communication protocol will be vital as it might affect the response times.

2.5 Mobile Device Limitations

Mobile devices have limited resources as compared to Personal Computers; they usually have processors that are around 100 MHZ or multiples of this whereas desktop PCs have processors that in the GHZ range. This eventually forces programmers to use efficient algorithms and data structures in order to produce applications with reasonable response times. This issue of limited resources is also addressed by Bowles [3] in a study conducted with undergraduate students creating a mobile phone game using efficient data structures and algorithms.

The quality of mobile device applications should also be considered when assessing limitations on mobile devices. Therefore the limitation on resources should not result in development of applications that are not robust. Umphress, et al. [11] address this issue by developing software which conforms to industry standards. The software developed by students in this paper was assessed using the Nokia OK Certification procedures. The Nokia OK Certification procedures are used to test if software is robust and conforms to the industry standards for wireless software development. This certainly helps in promoting the development of robust applications for mobile devices.

2.6 Summary

In this section we have discussed the different choices available for developing mobile device applications. After comparing the different environments available it was discovered that J2ME offers the best cross platform portability. There is also a greater range of mobile devices available. Therefore J2ME applications will reach a greater target group of users since it is widely supported. We also discovered that efficient algorithms and data structures can help in dealing with the limited resources available in mobile devices. As a result of this, robust applications can be developed by adhering to industry standards for wireless mobile devices and good programming practices which will clearly help in providing more functionality to mobile devices.

3. Design and Implementation

The Cellphone Shopper application was designed using a user-centred design approach. Users were involved from the beginning for the purpose of gathering system requirements and in the prototype stage.

3.1 System Overview

The system comprises of two layers: the front end and the back end. The front end is split into two interfaces with one being Cellphone-based and another one being Web-based. The back end is made up of a database and a Web application which is used by the two front end interfaces to retrieve and store data in the database. The communication protocol used by the two layers of the system is SOAP over General Packet Radio Service (GPRS) for the Cellphone interface and over the Internet for the Web Interface. The following figure illustrates the system overview and how the different components of the system are connected.

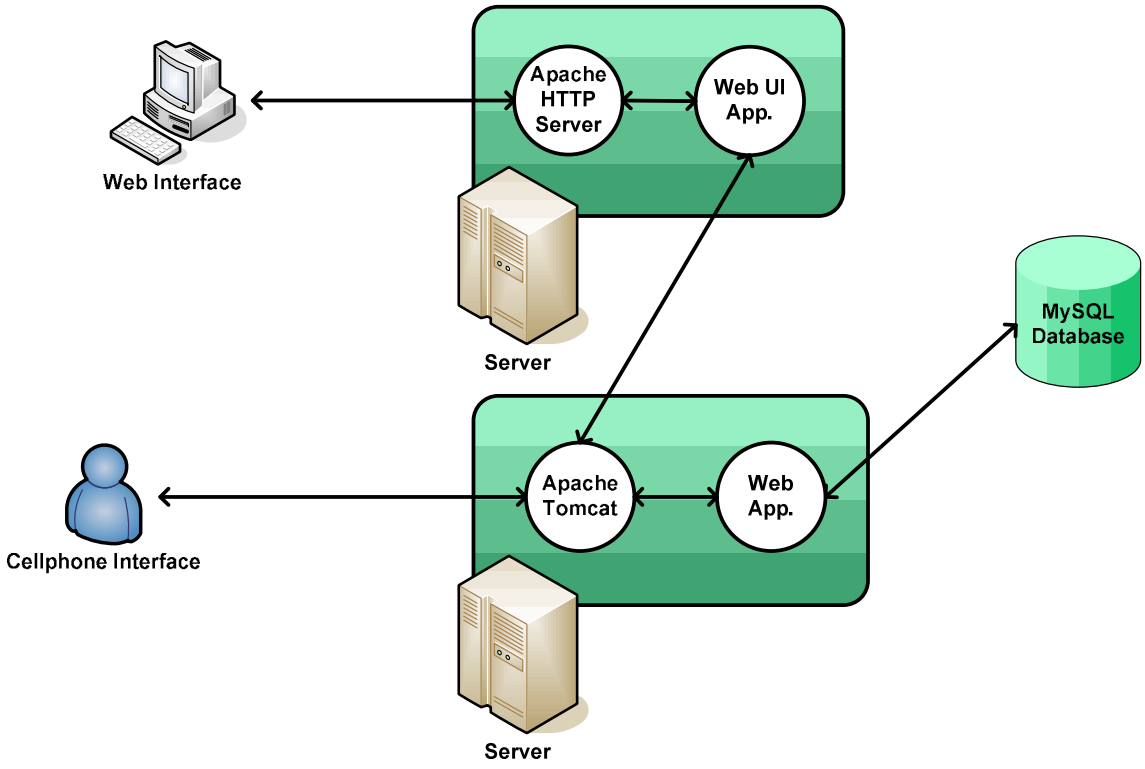

Figure 1: Cellphone Shopper System Design.

3.1.1 List Management

The system allows users to have multiple lists and they can add and delete these lists. When a list is created it is associated with a default shop which is assigned to items as the default shop from where the items are going to be purchased. The default shop is assigned to items which a user did not specify a shop when they were adding items to the list. The owner of the list can share a list with other trusted users who can be assigned access rights. These access rights include add and delete.

3.1.2 Lists

Users can perform certain operations on items in the list, like add, delete and edit. Whenever an item is deleted it is kept in a deleted list which is session-based meaning that it will only be available for that session only and after that it will be permanently removed from the list. If a user adds a new item and it does not exist they will have to specify the following details:

- The name of the item.
- Quantity.
- Category.
- The Shop to buy from. The default shop is always given.
- A note associated with the item for clear descriptions.
- The item can also be added to their favourite list.
- They can make it a private item and it will only be accessible by the owner.
- The item can be added to the list as an item they are not sure they really want to buy.

The only compulsory field needed is the name of item and the rest are optional. The following screenshot illustrates the interface used for adding new items.

Figure 2: Adding New Items

The shopping list can be viewed in different ways which are:

- Alphabetically.
- By Category the item belongs to.
- By Shop they Item is going to be purchased from.
- By Section of a particular shop, this is useful only when shopping so that the list will be arranged according to the layout of the shop.

The following figure shows the list being viewed alphabetically on the left, by category in the middle and by shop name on the right.

Figure 3: Viewing Items by Category or Shop Name.

The system supports creation of special lists which are:

- Favourites List - Users can add and delete items in this list.
- Previous Items - These are items that have been purchased in the past and the list is not editable. These items are added automatically when they are bought.
- Bought Items - This is a list of items that have been purchased during the current session.
- Deleted Items - This is a list of items deleted by a user in a particular session.

The following figure shows a screenshot displaying the special lists followed by the user's shopping lists. The user's shopping lists have a prefix of user.

Figure 4: Viewing Available Lists

3.1.3 Item History

When a user is doing their shopping, items in the list can be checked off as soon as they are purchased and they will be committed to the history of that particular user and they are viewable under the bought items and previous items lists. The following screenshot shows the users shopping list in shopping mode where the bought button is easily accessible in order to minimise the number of clicks need to check off items from the list.

Figure 5: Shopping Mode.

The bought list is composed of items bought within that particular session whereas the previous items list has all the items that have been purchased in the past. The previous items list is system generated; users can not edit or delete items in history. Users can only view the items they have added to other users' lists or items from their own previous lists. The items in previous lists can be used to add to new lists by simply selecting the desired items and then adding them to the current shopping list.

3.1.4 Shop Layouts

The system allows users to add their own shop layout that will be used to arrange their shopping list when viewing by shop while shopping. The system will provide a generic layout for supported shops and users can customise it for their personal use. New shopping layouts can also be submitted and will be available to other users of the system if approved by the administrators. The shop layout is represented as simple isles with all the categories that form part of that section. This is only available on the Web interface and can be used to arrange the list via the Cellphone interface.

3.1.5 Reminders

The system allows users to enter an item which can be used as a reminder. This feature can be used to manage utility bills like electricity, water, etc. When a user logs in, the system will remind them as soon as the specified date is reached.

When adding a new reminder the following details have to be provided:

- Name, Description and Date
- Notification Period - This is the number of days before the due date specified that the system has to remind the user.
- Set as item in list - This option will add the reminder as an item in the shopping list.

The following screenshot illustrates the interface used for adding new reminders.

Figure 6: Adding New Reminders.

3.1.6 Notifications

When a user logs in, the system will show a news feed informing the user about the changes made on the list since the last time the user logged in. These changes can be new items that have been added to the list, items deleted from the list and items that have been edited. If there are any reminders that are due, they will be shown as part of the notifications too. If they aren't any notifications the application will display the shopping list.

3.1.7 User Preferences

The user can specify their own preferences and details like their user name, e-mail, Cellphone number, password, etc. Users can use these preferences to customize their profile by selecting a list of their favourite shops which will be easily accessible when adding a new list or a new item. For example on the Cellphone application when a user wants to choose a shop, they will be presented with a list of their favourite shops. They can add a list of trusted users they want to share shopping lists with. This option is only available from the Web interface only.

3.2 Technology and Tools

Java 2 Micro Edition (J2ME) was chosen as the development environment for the Cellphone interface since it provides cross platform applications and there are more Java-enabled phones on the market. The platform independence provided by J2ME can therefore determine the success of an application due to the wider coverage of handheld devices. The device profile supported is MIDP 2.0 since it provides support for SMS, MMS, calendars, contacts and Bluetooth in the form of JSR-82. Even though J2ME applications are slower since Java applications run on the Kilobyte Virtual Machine (KVM), these applications still provide reasonable response times.

The IDE used for developing the application is Netbeans 5.5 with the Netbeans mobility pack 5.5.1. The mobility pack comes with the Sun Java (TM) Wireless Toolkit 2.5.1 which includes the default Sun Emulator. This tool was chosen as it has all the basic features that are supported by all Java-enabled phones. There are better vendor specific tools but some of these tools have features that are only supported by that particular manufacturer's phones which will limit the range of devices the application can run on. The Sun toolkit has all the features needed for packaging and successful distributing of the application.

3.3 Interface Design

Two groups of users were involved in the design of the Cellphone interface. Different types of users were interviewed ranging from people who do not use computers regularly to frequent users.

3.3.1 Gathering System Requirements

The first group of users was selected and interviewed to find out how they manipulate shopping lists. This group of users was comprised of people who do shopping for the whole family to users who shop for themselves only. Different shopping patterns and habits were discovered through this interview process. The users were also given a chance to suggest which features they would like to see in the system. The final list of features has been explained in detail above in section 3.1, with the addition of new features suggested by the users like reminders. The other features which were suggested include:

- Users should be able to send a message to other users.
- Statistics like which item was purchased the most.
- Users should be able to customise their own shop layouts.

The aim of the interviews was to design a system that will fit or easily be adaptable to the normal shopping routines of the users. As a result of these interviews some of the list management issues were resolved, like the way users write down and arrange shopping lists to the way they manage item information like name, description and quantity.

3.3.2 First Prototype Session

The second group was composed of mobile experts and they were involved in the actual design of the interfaces through a paper prototype session. The prototype was presented as a basic interface with limited functionality and was improved during the session in order to provide better functionality. The interface was designed in such a way that will suit the small screen and at the same time making it much easier to access the features of the system. The resulting interface was also assessed in order to remove any aspects of the design that did not conform to the design principles for mobile devices. Another aspect that was addressed in the session was designing an interface that can support both novice and expert users of the system. This was illustrated by having an interface with the same icons in the different interfaces which will match and best represent the tasks at hand.

Some of the suggestions made included having auto-complete for the item name and drop down boxes for item category and shop name. The multi-select screen for delete could also use strike-through text for deleted items. Options can be mapped to shortcut keys that are oriented or numbered as in the interface.

For example, while viewing the list by category name the right arrow can be used to select a category and the left arrow used to go back. Some of the suggestions made were addressed towards the functionality of the system like caching of events on the phone and sending the information to the server as a batch in order to minimise the cost of creating a connection every time a change is made within the application.

The following pictures show the paper prototype screens shown to users during the session. The first screen on the left below shows the suggested input layout for adding new items to the list. The second screen on the right shows the suggested list layout when viewing items by category and the method that can be used to show deleted items from the list.

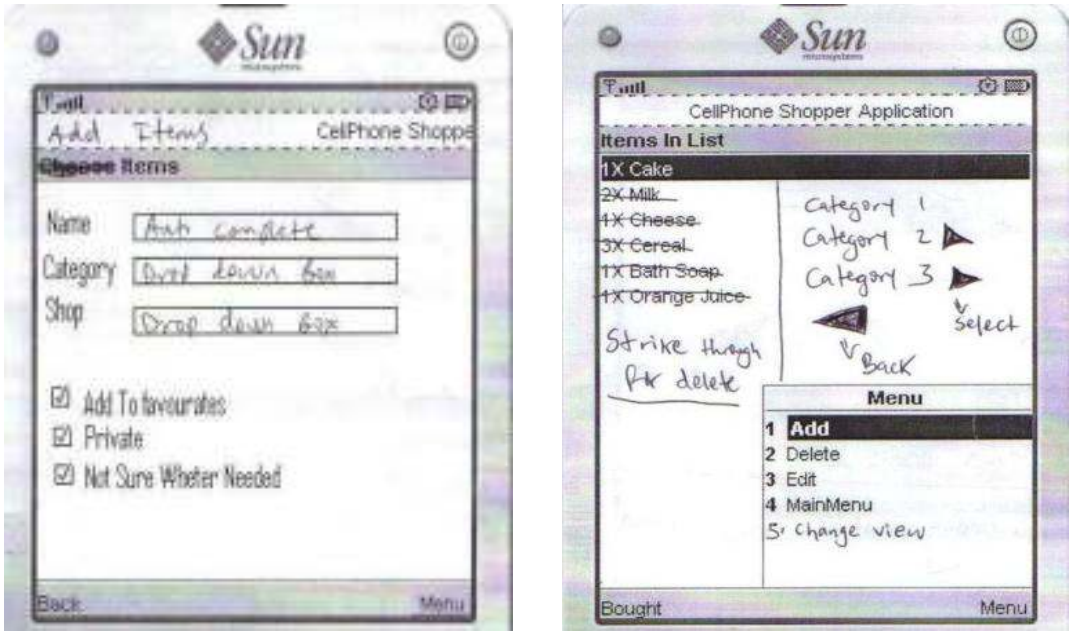

Figure 7: Paper Prototype Screens

The screenshot on the left shows the suggestions mentioned for adding new items with auto-complete for item name and drop down boxes for category and shop name. The screenshot on the right shows the suggested method for displaying deleted items by using strike-through. The navigation method that was suggested for selecting categories is also illustrated by using arrows on the right screenshot.

3.3.3 Second Prototype Session

The second prototype session included some of the expert users from the first session and new users. The aim of the second session was to assess the developed system since the first prototype session. The second session was conducted using the Sun Mobile Phone emulator and all the options available were ran at least once and assessed. The aim of the session was to improve on the usability of the system as a whole. Several screen layouts were rearranged in order to group similar functions together and at the same time improving on the usability of the system. Changes made included regrouping of menu options in the main menu and renaming some menu options in order to remove any ambiguities that may arise. The screenshot on the left below represents the menu layout presented at the session and the screenshot on the right represent the final layout after the changes were made.

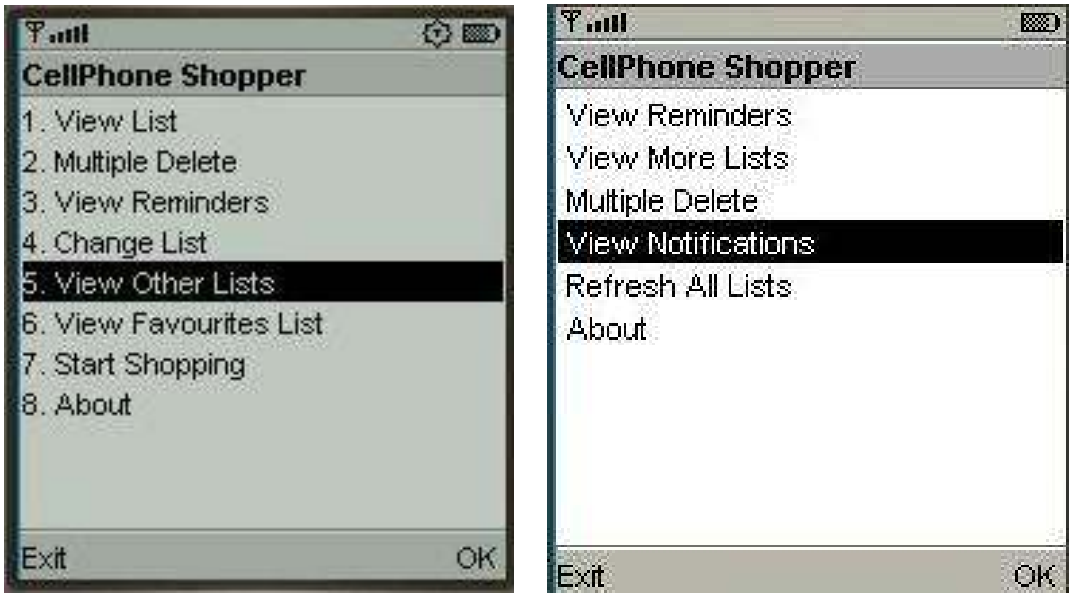

Figure 8: Second Prototype and Final Screens.

The changes made among other things, was to merge the options "Change List"-which is used to change the current user's shopping list and "View Other Lists"- which is used to display system generated lists. The two options were combined to form a new option that will be "View More Lists", which is option number two on the right. The "View List" option was removed since it did not serve a purpose any longer because it was used to select the last viewed user list. This allowed users to view all personal shopping lists and system generated lists in one screen removing the extra step that was needed to switch between the two.

The user is now able to view all the available lists as in the screen shown below and all the user's personal shopping lists were differentiated from system lists by having a prefix of user.

Figure 9: Viewing Available Lists.

The order in which fields are arranged while adding a new Item to the list was also changed. The item fields were rearranged based on importance or by putting first fields that are commonly used when adding new items. The screenshot on the left shows the order of fields for the prototype and the screen on the right shows the final arrangement adopted. The Item category field was placed second since it is needed when arranging lists based on shop layout and it is very crucial in defining the type of item being created. The item note field was placed last since users indicated that they use notes for items less frequently on shopping lists.

Figure 10: Viewing Available Lists.

Other suggestions included adding more options for a user while they are viewing items in the list. New options that were added to the menu include allowing users to change the ordering of items in the list, a quicker way to switch to another shopping list and a quicker way to switch to shopping mode. Users should not be shown list options that are not applicable, like delete items, edit items and start shopping if the list is empty. Options like add reminder were removed from the list and placed under the reminders menu. This was done in order to minimise the number of options available when viewing items since the number of options available were increasing beyond the desirable number that can be displayed on a small screen. Another option that was removed was "View Item Details" which was now mapped to the select button, so that when a user presses the select button the item details are displayed.

The following screenshots shows the prototype first in figure 11, followed by the suggested layout on the left in figure 12 with some items in the list and the suggested layout when the list is empty on the right.

Figure 11: Prototype Viewing Items in Lists.

Figure 12: Viewing Items in Lists.

The screenshot above on the left in figure 12 shows the list with all options available and on the right with only two options available. The four options that are not available are delete item, edit item, start shopping and change ordering which are all only applicable when the list has items. This will certainly improve the navigation through the interface since useless options have been omitted from the interface. Since all the options available are applicable, users will now be able to get a response from the application whenever an option is selected. The interactivity of the application is therefore improved by this method of presentation.

3.4 Communication Protocol

The communication protocol used by the front end and the back end of the system is SOAP over General Packet Radio Service (GPRS) for the Cellphone interface. The number of characters per item is reduced by using simple tags in the XML messages. The messages passed are stripped down XML to reduce the amount of traffic generated. This will certainly help to reduce the cost of transferring data from the Cellphone Application. The changes made on the application will be cached and then sent as a batch to further reduce the amount of traffic generated by the application. All requests made to server are through the GET request method for data retrieval and POST is used for sending updated data back to the server. All parameters in POST are encoded using application/x-www-form-urlencoding. A sample of XML communication for the different features of the application is shown below.

3.4.1 Authentication

Login

When a user inputs the login information the MD5 hash is computed for the password field and then sent to the server along with the username using a GET request. The parameters needed are ***UserName*** and ***UserPassword*** and the returned XML will be of the following format if the login operation was successful.

Format of XML:

```
<u id=UserId n=username f=firstName s=lastName" e=email c=phoneNumber />
```

Example:

```
<u id="3" n="HunterU" f="Graham" s="Hunter" e="hunter@test.com" c="55542024" />
```

Logout

When a user exits the application the server is contacted for logout purposes using a GET request. There is only one parameter needed which is the ***UserId*** and the returned XML will be of the following format.

Format of XML:

```
<id> userId </id>
```

Example:

```
<id>3</id>
```

3.4.2 Lists

Retrieving Available List Names

The application queries the server for available lists through a GET request with only one parameter needed which is the ***UserId***. If the user is logged in the response from the server will be XML which will be as follows:

Format of XML:

```
<Lists>
<list id=ListId uid=UserId n=ListName access=AccessRights s=ShopName  sid=ShopId/>
</Lists>
```

Example:

```
<Lists>
<list id="3" uid="3" n="newlist" access="111" s="Checkers, Rondebosch,  Riverside Centre" sid="1"/>
<list id="2005" uid="3" n="XList" access="111" s="Checkers, Rondebosch, Riverside Centre" sid="1"/>
<list id="2007" uid="3" n="Test List" access="111" s="" sid="-1" />
</Lists>
```

Retrieving Items for a Specific List

In order to get Items for a specific list the application uses a GET Request with two parameters which are the ***UserId*** and ***ListId.*** The server will then send an XML response with items that are in the requested list. The fields included in the returned XML are***: Name of Item, quantity, category, shop, a note, an indication of whether an item is private, an indication of the uncertainty of the item, username, user Id, item Id*** and ***date.***

Format of XML:

```
<Items>
    <I id=itemId n=itemName q=quantity c=category sh=ShopName sid=ShopId p=private
      u=uncertanity usr=username uid=userId dt=date>Note</I>
</Items>
```

Example:

```
<Items>
    <I id="1660" n="Beans" q="1" c="Butter" sh="Checkers, Rondebosch, Riverside Centre"
      sid="1" p="0" u="0" usr="HunterU" uid="3" dt="14-10-2007">Note</I>
   <I id="1659" n="Coffee" q="1" c="Bread" sh="PicknPay, Rondebosch, Riverside Centre"
      sid="2" p="0" u="0" usr="HunterU" uid="3" dt="14-10-2007"> Note </I>
   <I id="1658" n="Orange Juice" q="1" c="Cooldrinks" sh="PicknPay, Rondebosch, Riverside
      Centre" sid="2" p="0" u="0" usr="HunterU" uid="3" dt="14-10-2007"> Note </I>
</Items>
```

Adding New Items

For adding new items to the list a POST request method is used and all the parameters are encoded using "application/x-www-form-urlencoding". The parameters needed for a POST to work are ***ListId, UserId, ItemNameList, ItemCategoryList, ItemNote, ItemUncertain, ItemPrivate, ShopId*** and ***ItemQuantity.***

Adding New Items

When a user is editing items on the list the same procedure as for adding a new item is followed but now an extra parameter is added, which is the ***ItemIdList***.

Deleting Items and Adding to Special Lists

When adding an item to the favourites list, deleting an item or checking out an item after it is purchased; the POST will only have three parameters which are the ***UserId, ListId*** and the ***ItemIdList.***

Each parameter can be concatenated for any POST to represent multiple items so that multiple items can be sent to the server as a batch. The character used for concatenation is **"|"**, for example concatenated ***UserIds*** will be in this format:

UserId = **"**123|34|4|5|90**"**

POST Response

The POST response is an XML composed of item ids or an error message. The format of the XML will be as follows:

Format of XML:

<id> ***ItemIdList*** </id>

or

<error> ***error message*** </error>

Example:

<id> **3|4|5|6|7** </id>

or

<error> invalid ***UserId*** </error>

3.4.3 Reminders

Retrieving a List of Reminders

To retrieve a list of reminders a GET Request is used with one parameter which is the ***UserId***. The returned XML will be as follows***:***

Format of XML:

```
<Reminders>
    <r id=reminderId n=reminderName d=reminderDescription date=reminderDate
      p=reminderPeriod i=reminderItem />
</Reminders>
```

Example:

```
<Reminders>
  <r id="132" n="Electricity Bill" d="" date="14-10-2007" p="5" i="0" />
  <r id="131" n="Phone Bill" d="Pay at Shoprite" date="14-10-2007" p="5" i="0" />
  <r id="130" n="Water Bill" d=" " date="14-10-2007Pay at Shoprite" p="5" i="0" />
</Reminders>
```

Adding New Reminders

When adding new reminders the POST request method is used and all the parameters are encoded using "application/x-www-form-urlencoding". The parameters used are ***UserId, ReminderName, ReminderDescription, ReminderDate, ReminderPeriod*** and ***ReminderItem.***

Editing Reminders

Editing reminders works the same way as adding new reminders but with the addition of another parameter. The extra parameter needed is ***ReminderId***.

Deleting Reminders

When deleting reminders the POST request method is used and all the parameters are encoded using "application/x-www-form-urlencoding". The parameters used are ***UserId*** and ***ReminderId.***

POST Response

The POST response is an XML composed of reminder ids or an error message. The format is the same as in POST for items but ***ItemIdList*** is replaced by the ***ReminderId.***

3.4.4 Shops

To retrieve a list of favourite shops a GET Request is used with one parameter which is the ***UserId***. The returned XML will be as follows:

Format of XML:

```
<Shops>
        <Shop id=ShopId n=ShopName " l=Location c=City s=Suburb />
</Shops>
```

Example:

```
<Shops>
   <Shop id="1" n="Checkers" l="Riverside Centre" c="Cape Town" s="Rondebosch" />
   <Shop id="2" n="PicknPay" l="Riverside Centre" c="Cape Town" s="Rondebosch" />
</Shops>
```

3.4.5 Notification

To retrieve a list of notifications a GET Request is used with one parameter which is the ***UserId***. The returned XML will be as follows:

Format of XML:

```
<Notifications>
   <Updates>
        <List id=ListId n=ListName u=UserName type=Operation />
   </Updates>
   <Reminders>
        <r  id=reminderId  n=reminderName  d=reminderDescription  date=reminderDate
            p=reminderPeriod i=reminderItem />
   </Reminders>
</Notifications>
```

Example:

```
<Notifications>
   <Updates>
        <List id="3665" n="AutoList" u="Tshifhiwa" type="Add" />
        <List id="3665" n="AutoList" u=" Tshifhiwa" type="Edit" />
        <List id="3665" n="AutoList" u=" Tshifhiwa" type="Delete" />
   </Updates>
   <Reminders>
        <r id="132" n="Electricity Bill" d="" date="14-10-2007" p="5" i="0" />
        <r id="131" n="Phone Bill" d="Pay at Shoprite" date="14-10-2007" p="5" i="0" />
   </Reminders>
</Notifications>
```

3.5 Use Cases

Use Case 1: Login

Actor: List Owner

Description:

When a user runs the application the first screen displayed is the login screen where the user has to provide the username and password. If the login is not successful an error message will be displayed and the user will be asked to login again. If the login operation is successful the notifications screen is displayed which will have all the changes that occurred since the user's last login or the shopping list will be shown if there are no notifications.

Use Case 2: Adding a New Item

Actor: List Owner

Description:

When a user selects the ***add new item*** option a screen is displayed where the user has to input the details for the new item. The details for an item are name of the item, quantity, category, shop, note, indicate if item is private, an indication of whether the item should be added to favourites and an indication of the uncertainty of the item. The compulsory field needed is the name of item. The other fields can be left blank and the item will still be added if the name is provided.

Use Case 3: Editing an Item

Actor: List Owner

Description:

When a user selects the ***edit item*** option a screen is displayed where the current item details are displayed with all item fields editable. The user will then commit the changes and the edited item details are updated on the server. If the user does not commit the changes and chooses to go back the changes made are discarded and the user is taken to the previous screen.

Use Case 4: Deleting an Item

Actor: List Owner

Description:

The user selects ***delete*** while viewing the shopping list and the currently selected item or items are removed from the current list and inserted into the deleted list.

Use Case 5: Adding a New Reminder

Actor: List Owner

Description:

The user selects the ***add new reminder*** option and a screen is displayed where the user has to input the details for the new reminder. The details needed for a new reminder are name, description, date, period and an indication of whether an item is a reminder. When the user saves the details, the new reminder is then added to the reminders list.

Use Case 6: Editing a Reminder

Actor: List Owner

Description:

When a user selects the ***edit reminder*** option a screen is displayed with the currently selected reminder details with all reminder fields editable. The user will then save the changes made and the edited reminder details are updated on server.

Use Case 7: Deleting a Reminder

Actor: List Owner

Description:

The user selects ***delete reminder*** while viewing the reminders list and the currently selected reminder is then removed from the list.

3.6 Conceptual Class Collaborations

The following diagram shows all the modules of the system and the interactions involved between the modules. The different modules are then broken down to the underlying classes and discussed in detail.

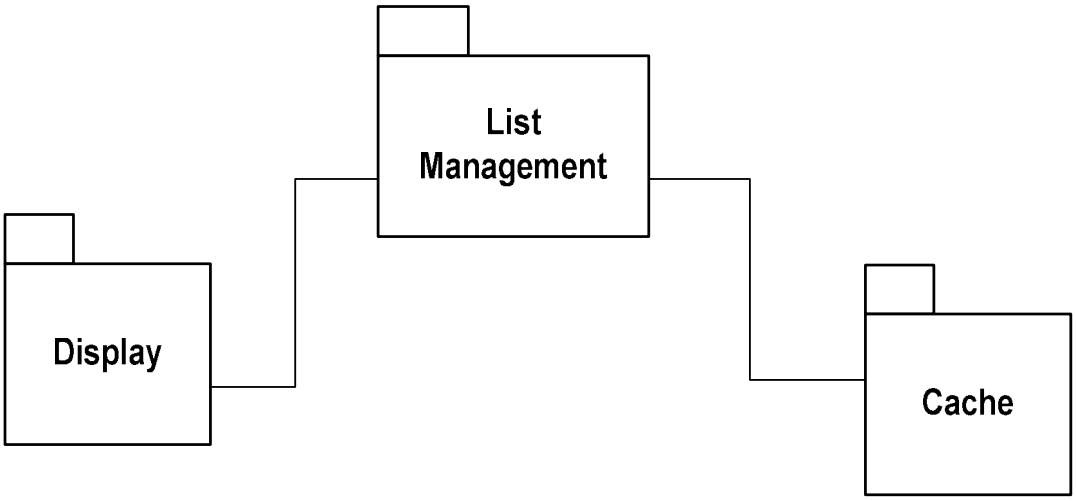

Figure 13: System Modules.

3.6.1 Display Module

The following diagram shows the interactions within the Display module.

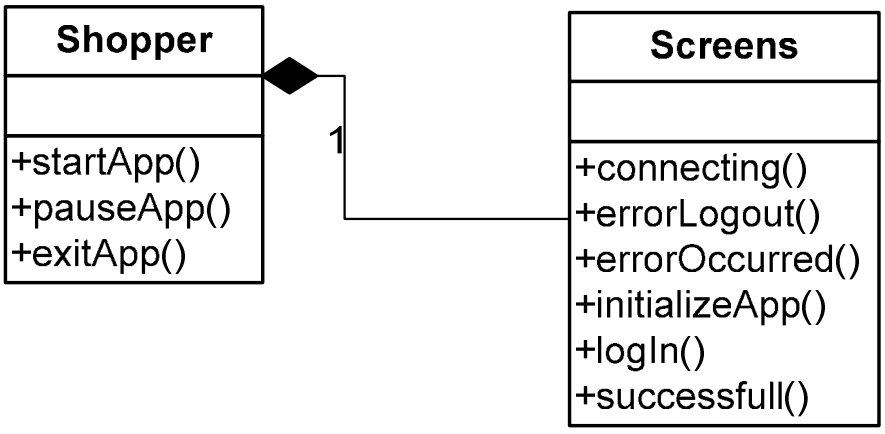

Figure 14: Display Module

3.6.1.1 Shopper Class

This is the Midlet class which is used to start up the application and uses the screens class to manage the screens displayed across the system.

3.6.1.2 Screens Class

This class manages all the screens that are displayed by the system and handles all the key inputs that a user makes while running the application. The access rights of the system are enforced here so that users can only view the options available to them using the access rights assigned to them. Users with all access rights for a particular list will be able to perform all operations available in the system.

3.6.2 List Management Module

This module handles all the users shopping list management like adding, editing and deleting of items and reminders from lists. The following diagram shows the interactions within List Management module.

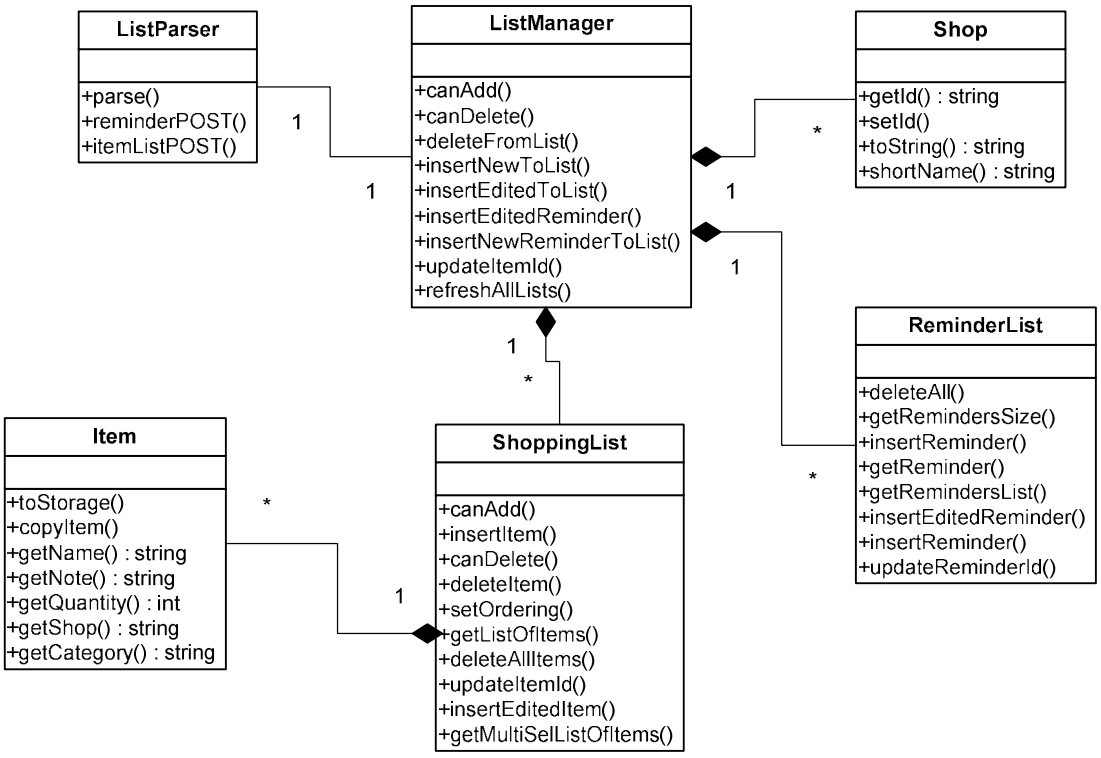

Figure 15: List Management Module.

3.6.2.1 Item Class

The Item class is used store an entire item's details. An item can either be a reminder or a normal shopping list item. All items belong to a certain shopping list and they are associated with a user who created the item.

3.6.2.2 Shop Class

This class handles favourite shops data available for a certain user. The shops available are classified based on their location. The shop location information available includes city, suburb and the actual place the shop is situated, like the name of the mall.

3.6.2.3 Shopping List Class

The shopping list class is used to store all the list information like name, default shop and the user who created the list. All items that belong to the list are stored in this class. The class handles all the operations that are performed on shopping lists like adding, editing and deleting items. Other functions like ordering of items alphabetically, according to category name and shop name of item are performed in this class.

3.6.2.4 Reminders List Class

All the users' current reminders are managed and stored by this class. All operations that can be performed on reminders are handled here. The operations handled include adding new reminders, editing and deleting reminders.

3.6.2.5 List Parser Class

The List Parser class is used for all communication with the server and XML parsing operations. All data is retrieved from the server by using simple GET requests and the returned XML is then parsed and data is passed to the List Manager Class. This class is responsible for all updates to the server using the POST request method to send data back to the server. The XML parsing is done using a pull parser which processes data as soon as it is read as compared to a model parser which stores the whole XML in memory first. The pull parser was chosen because it uses memory much more efficiently than a model-based parser.

3.6.2.6 List Manager Class

This class handles the authentication upon login and it is used to manage all the shopping lists that a user has. All operations that are performed on any lists are handled by this class. This class listens to parse events from the List Parser to update the current shopping list data and uses the list parser class to send any updates to the server.

This class manages the queuing of events in cache so that they can be sent the server as a batch in order to utilise the network efficiently. All changes that cannot be updated on the server due to network problems are then stored in cache or memory. All reminders and notifications are handled in this class.

3.6.3 Cache Module

This module handles all the internal storage of any changes made to shopping list data available. All similar events are cached and then sent to server as a batch.

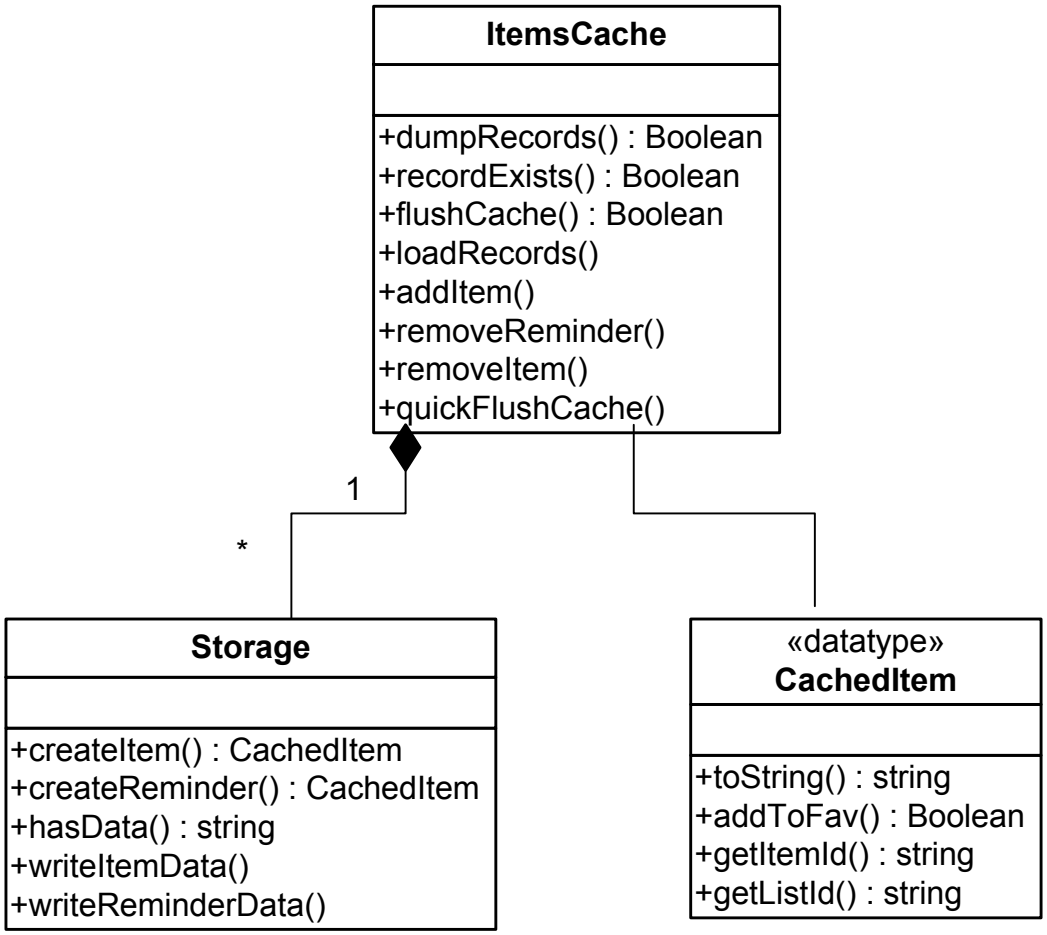

Figure 16: Cache Module

3.6.3.1 Items Cache Class

The Items Cache class is used to store all changes made on the shopping list that need to be sent to the server. The cache is kept consistent by removing stale copies and replacing them with the new copy that has the latest changes. If an item has been edited and it is being added to cache and the item already exists in cache the old item in cache is replaced by the new one, thereby keeping the latest changes in cache.

If the application is being closed and the cache has items that need to be sent to the server and the server is unavailable, the changes left in cache are then stored in memory. The changes stored in memory will be sent to the server upon next login.

3.6.3.2 Cached Item Class

This is an entity class used by the Items Cache to store data that needs to be updated on the server. This class represents a record in cache and all changes made to a certain item will always map to the same record stored in cache, thereby keeping the cache consistent and duplicate free.

3.6.3.3 Storage Class

This class handles all the data storage to internal memory. All items in cache upon exiting the application are written to memory by this class and then retrieved upon next login. To minimise the amount of data stored in memory only the edited fields are stored in memory. All records stored in memory are destroyed once they have been read back by the application.

3.7. User Interface Work Flow

When a user runs the application the Login screen is displayed where the user has to enter the username and password information that is used for authentication purposes. Once the user is logged in the next screen shown is the notification screen which displays all the changes made to the lists since the user's last session. Reminders which are due are also displayed as part of the notifications. If the notifications are not available the user's default list is then displayed. When viewing the available list the following options are available:

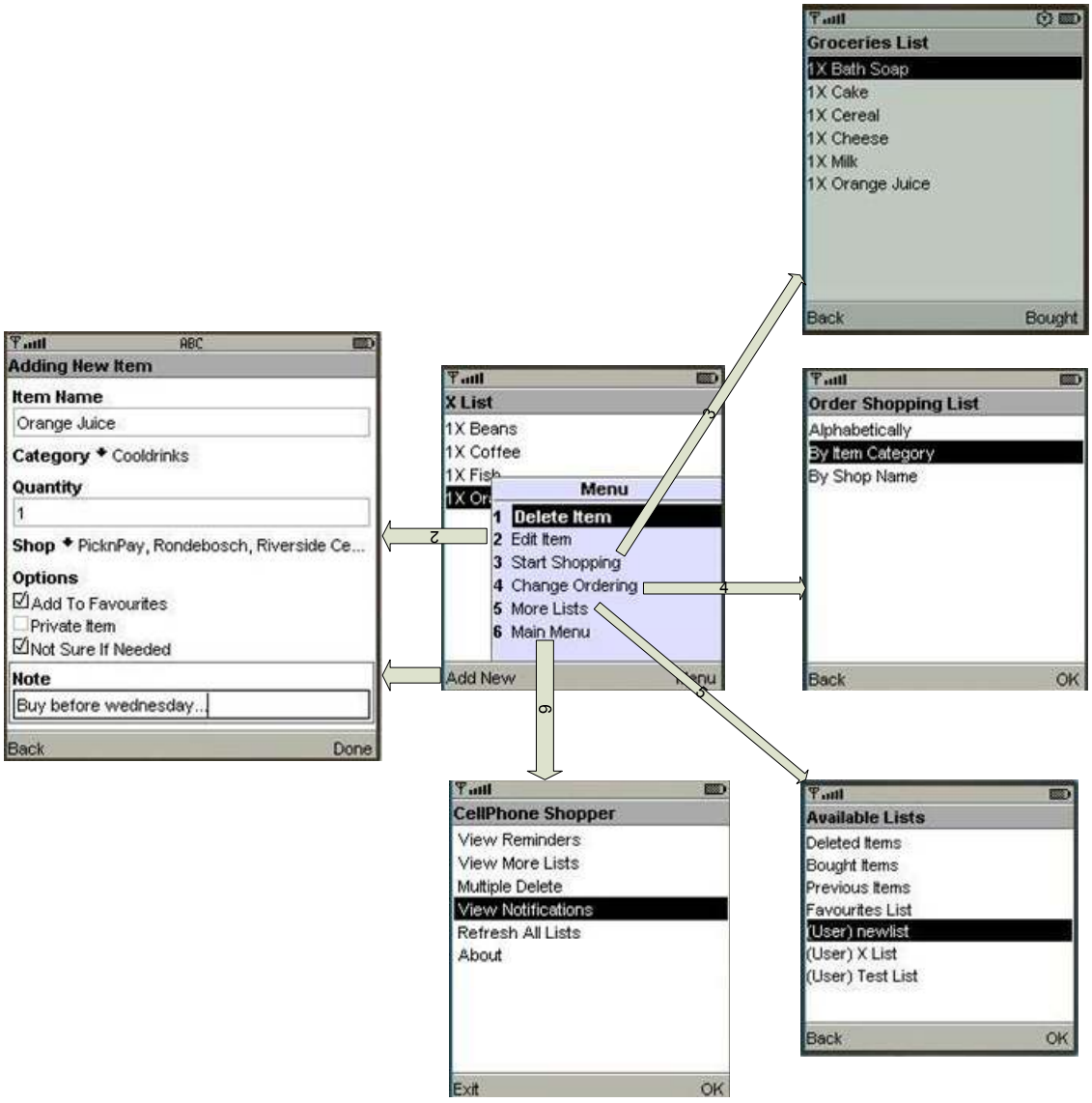

Figure 17: Options Available While Viewing List.

In Fig. 17, above the Screen at the centre displays the current list being viewed and the others screens represent the options that are available.

When a user is in the main menu there are several options available which are:

View Reminders – Where users can manage and view all the available reminders.

View More Lists – This option displays all the lists available for a particular user.

Multiple Delete – This option allows users to select multiple items and delete them as batch.

View Notifications – This will display all the changes made to the list ever since the last session and all reminders that are due.

Refresh All Lists – This option reloads all list data from the server.

The following figure illustrates the screen flow for the options explained above.

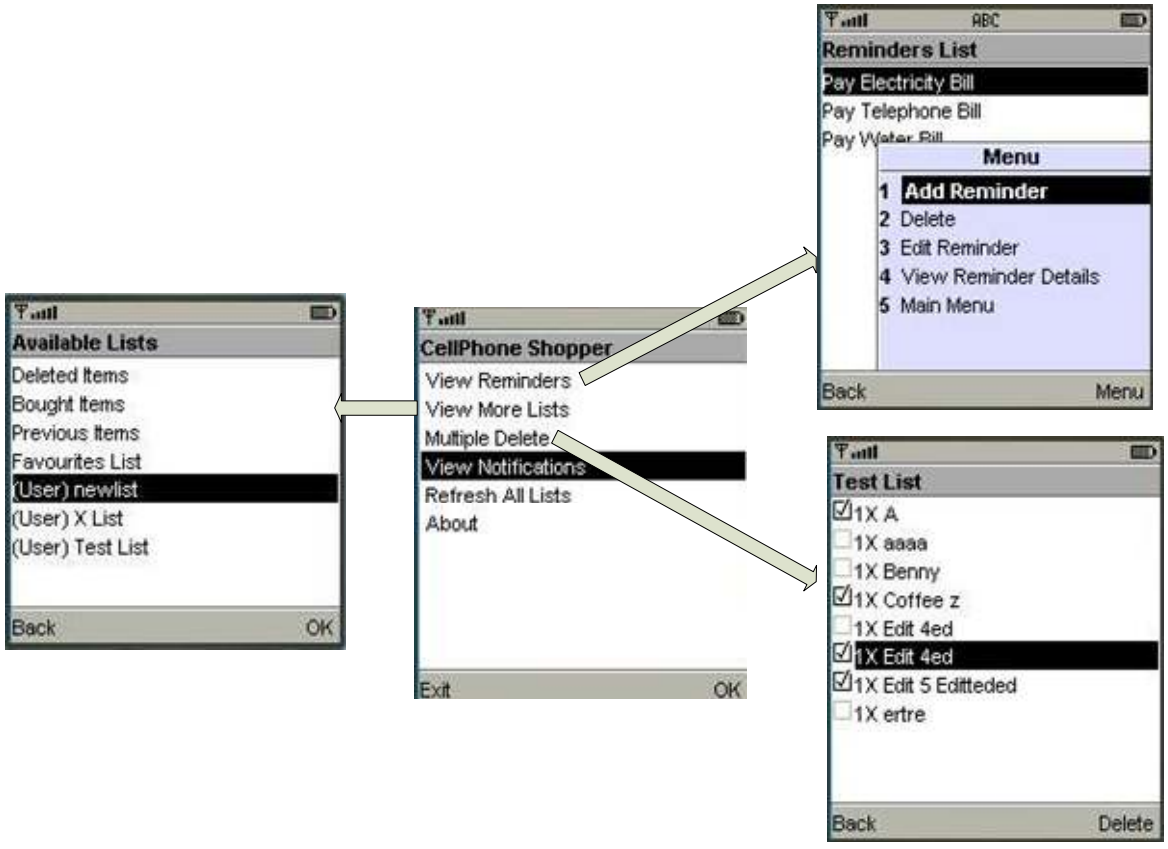

Figure 18: Options Available in Main Menu.

4. Evaluation and Testing

Several tests were conducted on the system in order to detect any errors and bugs that are in the system. The aim of these tests was to make sure that the quality of the software being developed is up to standard. Black Box and White Box testing methodologies were conducted in order to fully test the system as whole. These tests were done before the final user studies were conducted in order to make sure that the system is robust and does not affect the usability test conducted with users. After the system was deemed reasonably functional the final user studies were conducted in order to evaluate the usability of the system.

4.1 System Testing

The system was developed using an iterative approach where modules were integrated to the system one at a time. After a module was integrated into the system, tests were conducted in order to identify any problems that may arise from the integration process.

4.1.1 Black Box Testing

Black box testing was used to test the functionality of the system to make sure that all the requirements specified are satisfied by the system. Data used for some of the tests was carefully selected in order to assess the correctness of the system. Incorrect data was used to determine if the system would be able to handle the data accordingly. As a result the incorrect data could not be accepted by the system since input data fields were restricted to certain data types. The system was able to pass all these tests and as result passed the testing phase. The following test cases elaborate more on the different tests that were conducted.

Test Case 1: Adding a New Item

Test Conducted on: Phone

Method: A new item was created with all the item details specified and the field indicating that the item should be added to favourites was ticked.
Results: The item was added to the list and was also inserted to the favourites list.

Test Case 2: Editing an Item

Test Conducted on: Phone

Method: An item in the list was selected and all the fields were modified.

Results: The item was updated successfully.

Test Case 3: Editing an Item 2

Test Conducted on: Phone

Method: An item in the list was selected and some of the item details were modified but the name of the item was deleted and left blank.

Results: The application gave an error and requested the user to enter a name for the item. The system kept on returning to the edit screen when the user tried to update item details without giving a valid item name.

Test Case 4: Deleting an Item

Test Conducted on: Phone

Method: An item was selected and then the delete option was chosen from the menu.

Results: The item was successfully removed from the list.

Test Case 5: Deleting Multiple Items

Test Conducted on: Phone

Method: Several items were selected from the shopping list and then deleted.

Results: All the items selected were successfully removed from the list and were inserted to the deleted items list.

Test Case 6: Checking out Items

Test Conducted on: Phone

Method: Several items were selected as purchased items in shopping mode.
Results: All items selected were removed from the list and inserted to bought and previous lists successfully.

Test Case 7: Adding a new Reminder

Test Conducted on: Phone

Method: A new reminder was created with all the reminder details specified.

Results: The new reminder was added to the application successfully.

Test Case 9: Deleting a Reminder

Test Conducted on: Phone

Method: A reminder was selected and then the delete option was chosen.

Results: The reminder was removed from the application successfully.

4.1.2 White Box Testing

White Box Testing was conducted mostly to assess how the Cellphone application will deal with network problems. These tests were conducted to cover the different scenarios that are handled by the cache and the phone's internal memory when updates can not be sent to the server. This was achieved by using the application when there was no network or when the phone did not have airtime. These tests were conducted in order to fully test all the cases that were handled by the Cache module for storing updates on the phone's internal memory. White Box testing was used mostly to test the final application in order to fully test all the available paths of the system after the integration process. The Application was able to pass all the tests conducted to address the paths that wouldn't normally be taken when using the application.

Test Case 1: Sending Updates to Server from Cache

Test Conducted on: Sun Emulator

Method:

The computer network cable was unplugged so that the computer will not have access to the Internet. This was done in order to make sure that the application will not be able to contact server for the purpose of updating the lists. Several items were created, others were edited and some items were deleted. After the application failed to contact the server a couple of times the network cable was reconnected.

Results:

The application was able to send all the updates to the server successfully after the network cable was reconnected.

Test Case 2: Sending Updates to Server from Phone Memory

Test Conducted on: Phone

Method:

The phone was left with 2c airtime and several changes were made to the list until the phone did not have anymore airtime. When the phone was unable to communicate with the server the application was closed while they were still updates stored in cache. This was done in order to make the application save all the changes to the phone memory. The application was then restarted after the airtime was recharged.

Results:

The application was able to read all the updates stored in memory and they were sent to the server successfully.

4.2 User Testing

Users were involved in the final testing of the developed interface. This was done in order to evaluate the usability of the interface. Usability testing was conducted with nine users, where three were considered to be experts in the field of mobile devices. Fours users were considered to be novices and they did not really have any experience in using mobile applications. The other two were average users who occasionally used mobile applications.

The tests were performed on a Nokia 6288 and each user was asked to perform a set of pre-determined tasks on the application. These tasks covered all of the core features that are available on the mobile application. For each new screen that a user encountered they were asked to explain what they think the various components on the screen are for. Users were asked to perform the following tasks on the system:

1. Manipulating current Lists
   - Add a new Item
   - Edit an Item
   - Delete an Item

2. Changing ordering of Items in the List

3. Checking out Items

4. Managing Reminders
   - Add a Reminder
   - Edit a Reminder
   - Delete a Reminder

4. Managing Previous Lists
   - Add an Item From Previous Lists
   - Add an Item From Favourites

After completing the tasks the users were asked:

1. To comment on what they think about the interface.
2. What they don't like on the interface.
3. What they think could be improved.
4. Users were also asked to rate the difficulty of the different tasks on a scale of 1 to 5 where 1 was very easy and 5 was very difficult.

4.2.1 Critical Incidents

The following incidents were observed while users were performing the tasks:

- When adding new items most users did not understand what the note field was used for and they spent too much time scrolling through the list of categories in order to find the desired category.
- Some users could not figure out where to go to add new reminders while they were viewing the list until they were in the main menu.
- Most users did not understand what the notification period was when adding reminders. Some users thought it was the number of days that the system will notify you after the date entered. The main purpose of this option was to let users choose the number of days they want to be notified before the deadline and not after the date has passed.
- Few users did not understand how the **"Change Ordering"** option works. One of the users thought they could order items in the list manually based on priorities instead of ordering the list by category or shop name.
- Some users could not figure out what happened to the list when they changed the ordering of items in the list to category and the items disappeared and there were category names in the list.

  The following figure illustrates the transition that occurred.

Figure 19: Changing to Category View.

- All the users did not understand the difference between the system generated lists and the user lists even thought the user lists had a prefix of user on the name. Figure 20 below shows the Screen that all users had a problem with.

Figure 20: Available Lists.

4.2.2 User Feedback

The users were asked to rate the difficulty of the each task on a scale of 1 to 5 where 1 was very easy and 5 was very difficult. The average score was computed and the results for each task were then analysed.

Adding new items, deleting items and editing items were rated as being easy. Adding new reminders was rated as being average whereas editing and deleting reminders were rated as being easy. The main reason for adding reminders being rated as average was due to the fact that in order to add a new reminder, the user has to go to the main menu first. This was not apparently easy as the users viewed the shopping list first after the first time they logged into the application. Most users were not aware that reminders are in the main menu. After they found where reminders were situated it was easier for them to perform edit and delete operations as they are performed the same way as in manipulating shopping list items.

Changing currently viewed list was rated as being average because users did not understand the notion of system lists and user lists as already explained in 4.2.1 under the critical incidents section. Rearranging the order of items in the shopping list was rated as being between easy and average. The rating may have been affected by the users perception of what they thought the option was used for as already explained in 4.2.1 under the critical incidents section.

4.2.3 Discussion

Users gave conflicting feedback because some users thought that there were some features of the system that increased the complexity of using the system like the use of reminders. Other users thought that reminders were useful and it is a feature that they would like to use. The system in general was easy to use and users could perform all the basic operations like add, edit and delete without any difficulties. The problem that arose from viewing user lists and system lists was sorted out once users understood how the lists were created beforehand on the Web interface. Users seemed to like the idea of system generated and user lists afterwards as they thought it provided a quicker way for switching between these lists. Although users were not sure what was going on sometimes, they were able to figure out what was happening without any explanation even though there was hesitation before proceeding. They weren't any flaws found in the way the interface is designed that will hamper the usability of the application.

5. Conclusion

The Cellphone Shopper application was developed for the purpose of managing shopping lists that can be shared among several users. The developed application meets most of the requirements that were specified and allows users to share shopping lists efficiently. Here is the list of problems that the project was supposed to address:

- Having a common location where the list is shared in a household.
- Solving confusion that may arise when changes are made to the list like adding new items or removing items and no one knows who made the changes.
- Misplacing the shopping list or the list getting lost.
- Leaving the shopping list behind while going to do shopping.
- When doing shopping some items might not be described in a way that will distinguish them from other brands.
- Arranging shopping lists in a manner that will minimise time spent while shopping.

All of these problems mentioned above were solved by using efficient methods and usable interfaces. Some of the problems solved include being able to access shopping lists from any location since it is located on a central server and at the same time solving the problem of lists being misplaced or getting lost. Users are allowed to put notes on shopping list items in order to explain the purpose of the item. Optimisation of shopping time was also achieved by using shop layouts to arrange items in lists. Although they are minor synchronisation issues involved when users are manipulating lists on both the Cellphone and Web Interface simultaneously, the project was a success. The system was developed using a user-centred approach in order to come up with a usable design. The user studies conducted showed that users were able to perform most operations without any difficulties. The user interface developed can be considered to be easy to use and conforms to the design principles of usable interfaces. The project as a whole was a success as it provided an efficient way of managing shopping lists.

6. Future Work

6.1 Application's Views

- Navigating through the item categories list can be improved by allowing users to enter prefixes of a category and then filtering the list of categories to show only the categories that have the given prefix.
- Navigating through favourite shops can be improved too by allowing users to enter a substring and then filtering the list of shops by only showing shops that have the given text in the name instead of using it as a prefix.
- Previous items can be viewed in different ways in order to filter out the items much more efficiently to suit the user's needs. Additional views that can be supported include:
    - View by user who added the Item.
    - View by Shop the item was purchased from.
    - View items purchased between two dates.
    - Statistics like most popular items.

6.2 List management

- Adding the ability to create and manage the list information from the Cellphone application, such as preferences and who can access specific lists.
- Providing shop layouts from the mobile application and displaying the actual map of the store with item categories integrated in the image.
- The Cellphone application can support taking a picture or a video of barcodes and then use the data in the barcode to determine which item was scanned. This feature can be used to add new items to the list or to check out items after they are purchased.
- Bluetooth synchronisation can be implemented so that the Cellphone application can communicate with server to retrieve or update lists. This will be cost effective as Bluetooth does need the phone to have any airtime in order to work.
- The system can also provide product suggestions based on users' list history like suggesting to a user to add an item they usually purchase.
- State management of purchased items can be supported by the system and the user will be informed of products that going to expire or products that have already expired.

6.3 Communication

A note can be sent to another user through an SMS from the Cellphone interface or an email if the Web interface is being used. These notes can be used to tell another user to do shopping on their way home or to send any messages. The system can support the ability to notify users whenever changes are made to the shopping lists through an SMS or email.

6.4 Community Features

- The system can also add community features like the ability to inform others about various events, like items being out of stock in a particular shop.
- Specials and discounts from certain shops could also be among the features supported in community features.

6.5 System Interoperability

The Cellphone shopper system can interoperate with other systems in order to support online shopping by means of sending the user's shopping list to another shop's system like Pick 'n Pay or Woolworths. The items ordered will then be delivered by the shop to the users address.

References

[1] Bennett, J. M., Armstrong, M. L., and Gupta, S. 2004. A Message Board Client for handheld devices. In *Proceedings of the 42nd Annual Southeast Regional Conference* (Huntsville, Alabama, April 02 - 03, 2004). ACM-SE 42. ACM Press, New York, NY, 17-18.

[2] Black, J. T. and Hawkes, L. W. 2006. A prototype interface for collaborative mobile learning. In *Proceeding of the 2006 international Conference on Communications and Mobile Computing* (Vancouver, British Columbia, Canada, July 03 - 06, 2006). IWCMC '06. ACM Press, New York, NY, 1277-1282.

[3] Bowles, J. B. 2007. Cell phone games for a CS2 data structures course. In *Proceedings of the 45th Annual Southeast Regional Conference* (Winston-Salem, North Carolina, March 23 - 24, 2007). ACM-SE 45. ACM Press, New York, NY, 300-303.

[4] Davis, V., Gray, J., and Jones, J. 2005. Generative approaches for application tailoring of mobile devices. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2* (Kennesaw, Georgia, March 18 - 20, 2005). ACM-SE 43. ACM Press, New York, NY, 237-241.

[5] J2ME. Java ME Technology. Retrieved July 05, 2007 from Sun developer Network. http://java.sun.com/ javame/ technology/index.jsp

[6] Ludford, J. P., Frankowski, D., Reily, K., Wilms, K. and Terveen, L. Because I Carry My Cellphone Anyway: Functional Location-Based Reminder Applications. Proceedings of the SIGCHI conference on Human Factors in computing systems: Everyday use of mobiles. April 22-27, 2006, 889 – 898

[7] Paul Coulton , Omer Rashid , Reuben Edwards , Robert Thompson, Creating entertainment applications for cellular phones, Computers in Entertainment (CIE), v.3 n.3, July 2005

[8] Pick 'n Pay Home Shopping. Retrieved July 05, 2007, from Pick 'n Pay. http://www.picknpay.co.za/shopping.html

[9] Stefan Seedorf , Axel Korthaus , Markus Aleksy, Creating a topic map query tool for mobile devices using J2ME and XML, Proceedings of the 4th international symposium on Information and communication technologies, January 03-06, 2005, Cape Town, South Africa

[10] Tarnacha, A. and Maitland, C. F. 2006. Entrepreneurship in mobile application development. In *Proceedings of the 8th international Conference on Electronic Commerce: the New E-Commerce: innovations For Conquering Current Barriers, Obstacles and Limitations To Conducting Successful Business on the internet* (Fredericton, New Brunswick, Canada, August 13 - 16, 2006). ICEC '06, vol. 156. ACM Press, New York, NY, 589-593.

[11] Umphress, D. A., Cross, J. H., Jain, J., Meda, N., and Barowski, L. A. 2004. Bringing J2ME industry practice into the classroom. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA, March 03 - 07, 2004). SIGCSE '04. ACM Press, New York, NY, 301-305.

[12] Wesson, J. L. and van der Walt, D. F. 2005. Implementing mobile services: does the platform really make a difference? In *Proceedings of the 2005 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on IT Research in Developing Countries* (White River, South Africa, September 20 - 22, 2005). ACM International Conference Proceeding Series, vol. 150. South African Institute for Computer Scientists and Information Technologists, 208-216.

[13] White, J. 2001. An introduction to Java 2 micro edition (J2ME); Java in small things. In *Proceedings of the 23rd international Conference on Software Engineering* (Toronto, Ontario, Canada, May 12 - 19, 2001). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 724-725.

[14] Woolworths online shop. Retrieved July 05, 2007, from Woolworths. http://www.woolworths.co.za/caissa.asp

[15.] MXit, Retrieved July 13, 2007, from MXit
http://www.mxit.co.za/

[16.] Google Mail, Retrieved July 13, 2007, from Google Mail
http://www.gmail.google.com/

[17.] Go Talk Mobile, Retrieved July 13, 2007, from Go Talk Mobile
http://www.gotalkmobile.com/

Appendix A: Questionnaires

Questionnaire used by the Interviewer to ask Questions to Users for requirements gathering purposes.

Cellphone Shopper Project | Questionnaire

Interviewee(s):

Occupation(s):

1. Current Behaviour

- Who does the shopping?
  → Is the shopping done at more than one store?

- Your shopping list:
  → What is it written on?
  → Where is it kept?
  → Who makes the list?
  → Can anyone add to it?

2. Tell the interviewee(s) about Cellphone Shopper – its features.

(Multiple users can access the list, private items, item annotation, "regular items" list, list history, shop layouts, in-store shopping routes, ability to check off items as you buy, access via Cellphone and Web interface.)

3. Talk about the system and how they would use it.

- Would you mark off items as bought via the Cellphone interface as you buy them, or would you mark them off later via the Web interface?
- How would you like your list arranged on the Cellphone as you shop: alphabetically, in order of importance, as you would find items as you walk through the shop (by section), or some other way?
- Would you like to be notified when someone is doing the shopping and/or when someone changes you list (adds to, deletes from, etc.)?

- Would it be helpful to be able to add items that are hidden from others who can view the list? (E.g., private items or a birthday gift for someone who can view the list.)
- Would you like the system to be able to automatically add items you regularly buy to your shopping list?
- Would it be helpful to have a history of your past lists?

4. General

What do you think about the system – are there any other features you'd like to see in it?

This was used to evaluate the usability of the interface.

Cellphone Shopper Evaluation Form

Interviewee(s):

Occupation(s):

A. How a person perceives a screen the first time it is viewed

B. Ask what they think is the function of each screen element

C. Users completing a set of Tasks

1. Manipulating current Lists
   - Adding a new Item
   - Editing an Item
   - Delete an Item

2. Changing ordering of Items in the List

3. Checking out Items

4. Managing Reminders
   - Adding a Reminder
   - Edit a Reminder
   - Delete a Reminder

D. Questions Asked after the study.

1. Users were asked to comment on what they think about the interface
2. What they don't like
3. What they think could be improved upon.

Users were the asked to fill the following questionnaire.

For the following questions, please circle the number which best represents your level of difficulty experience while performing each task.

1. Adding an new Item

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

2. Editing Items

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

3. Deleting Items

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

4. Adding a new Reminder

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

5. Editing Reminders

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

6. Deleting Reminders

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

7. Changing currently viewed list

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult

8. Rearranging the ordering of items in the list.

1	2	3	4	5
Very Easy	Easy	Average	Difficult	Very Difficult