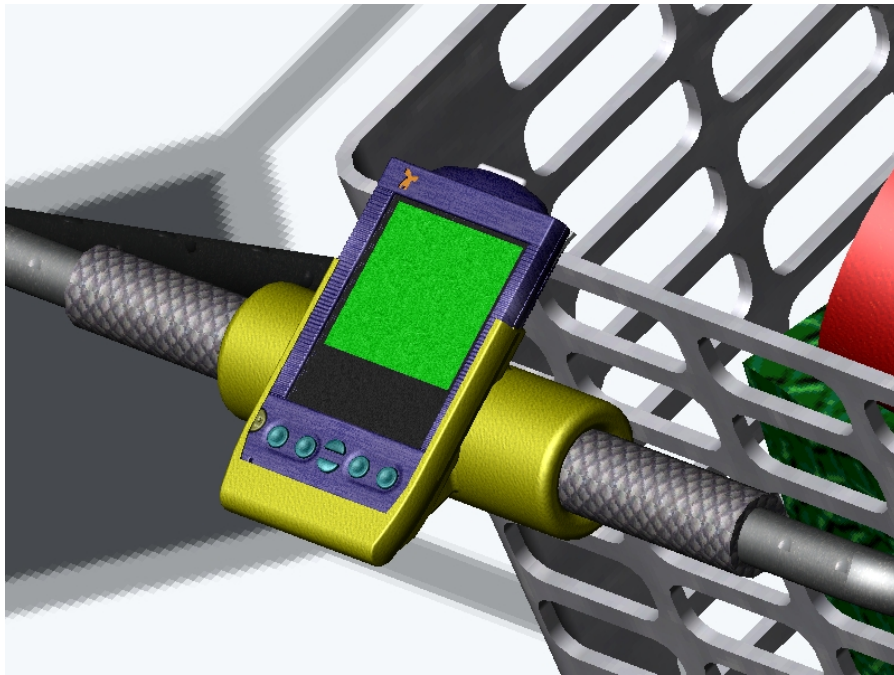


Smart Information Systems

Presents...



AWARETM

Developed for Interea Inc by Team Five:

- Jeremy Michalek
- Miguel Adler
- Sara Donner
- Yong Gyun Ghim
- Chang Yoon Yang

EXECUTIVE SUMMARY

The following is a report detailing the research and design of a 'global, Internet-ready product' for a computer software and peripherals company.

The product to be developed is a system to personalize information about products bought in a supermarket. The vast amount of information available in the modern world and the increasing number of medical or environmental concerns of shoppers have made shopping increasingly confusing with many complex decisions to be made. The product developed here aims to give fast, direct access to information tailored for and by the individual.

To enhance the global aspect of the product, primary and secondary marketing regions were chosen, and the product was adapted for each of these based on market research which is detailed in the report. The primary market chosen was the USA and the secondary market the UK.

Central to all regions is a collection of databases and query management systems called the Smart Information System. This stores and updates information from supermarkets, manufacturers and consumer information groups. Access to information about a specific product is via the Internet, using a variety of different sub-products. Each of the sub-products enables the user to set their own user profile, and to scan the barcode of a product to access information about it.

The sub-product chosen for development in the USA is called 'AwareTM'. It is a module for the Handspring Visor PDA, and incorporates software enabling the user to set a profile and to scan barcodes of products. Many Americans already own and use such PDA's and are generally willing to embrace new technology, so this is a suitable product for the region.

The database has been designed and prototyped, as is described in the report. The software has been programmed and the module designed. CAD drawings of the module and of an attachment to hold the PDA on a shopping cart have also been included.

For the UK, a different product would be used. This would be owned by the store and permanently attached to the shopping trolley. It would have the same functions as the USA model. Users would log in at the store to access a user profile which they set on the website using their home PC. This is appropriate for the UK where fewer consumers own PDA's or are willing to spend money on unfamiliar technology.

The Smart Information System has been carefully researched and designed and it is believed that it could be successfully marketed in many regions of the world using product variations customized to local needs.

Contents:

1. Introduction
2. Market Study & Analysis
 - 2.1 Project Selection
 - 2.2 The Market
 - 2.3 Selection of Primary and Secondary Markets
3. 'Globality' & Adaptability
4. Product Development for the USA
 - 4.1 Implementation
 - 4.1.1 Scanner Module & Wireless Internet Connection
 - 4.1.2 PDA Software
 - 4.1.3 Smart Information System: Database Structure
 - 4.1.4 Ergonomic Attachment
 - 4.2 Production
5. Plans for Launch in the UK
6. Conclusion

Appendices:

- Appendix 1 – Uses and Limitations of the Internet
- Appendix 2 – Initial Regional Research
- Appendix 3 – Consumer Survey
- Appendix 4 – Consumer Survey Results
- Appendix 5 – Regional Consumer Concerns
- Appendix 6 – Regional Consumer Attitudes
- Appendix 7 – General Regional Observations
- Appendix 8 – AwareTM Software Code
- Appendix 9 – Database Operation
- Appendix 10 – Letter To Potential Information Sources
- Appendix 11 – Potential Information Sources
- Appendix 12 – Sample Reply From Potential Information Sources
- Appendix 13 – SQL Code for Database Queries
- Appendix 14 – Observation of Shoppers
- Appendix 15 – Symbol CSM-150 Scanner Module Spec Sheet

Smart Information System: Product Review

1. INTRODUCTION

The Company: Interea Inc. is a San Diego based company, traditionally producing both low cost computer peripherals and sophisticated software solutions. At present, struggling in the current industry slowdown, the company wants to exploit its expertise in the area of synergising hardware and software, using the great resource of the Internet. To this end, Team Five was formed to investigate and product prototype for a new business venture that makes expedient use of these technologies.

The Brief: The brief is to produce a product that is both global and Internet-ready, both of which terms require definition. In the following, a global product is taken to be a product which, some adaptation to meet differing local needs, can feasibly be marketed in more than one region of the world. An Internet-ready product shall be defined as one that not only makes use of the Internet, but answers a real need that is present in society, a need that is best answered by using the Internet. To be truly Internet-ready, the product must have a user interface suitable for non-experts, and must be reliable and fast. Thus careful consideration must be made of the current capabilities and limitations of the Internet.

The Team: Team Five consists of five members, two from Seoul, Korea; two from Michigan, USA; and one from Oxford, UK. The team has a wide spread of skills, with some members specialising in design, some in market research and some in analysis. This spread has ensured that high priority was given to meeting the globally differing needs of consumers, as well as to the engineering development of the product.

Specific skills of the team include Product design and Prototyping, Mechanical design, CAD, CAM, CAE, Programming in C, C++, Basic, Fortran, HTML, JAVA and LISP, Mechatronics, Optimization, and Industrial Design. Expedient use has been made of each member's skills, and careful attention paid to differences of opinion arising out of our differing nationalities, in order to maximize the global usefulness of the product.

The Product: As an increased number of different products can be found in every corner of the world, the decision making process for purchasing any good gets more and more complex. With so many options to choose from and not enough time to gather the right information, arises the need for a fast and reliable way to conveniently find this information during the time of a purchase.

A sub-product of the SIS (Smart Information System), Interea's new business venture, AWARE™ is an expansion module for the Handspring Visor PDA (see next paragraph). It is one of several products that will enable consumers worldwide to access the SIS, initially for information on food products sold through supermarkets and convenience stores.

The PDA: A PDA (Personal Data Assistant) is a portable computer designed to act as an organization, note taking, and communication device. PDAs are small in size, and they generally use high-tech interfaces such as touch screens, hand writing recognition, or miniature keyboards. Each PDA runs on an operating system. PalmOS is one type of operating system used by several classes of PDAs including Handspring, PalmPilot, and IBM Workpad. Alternative PDA operating systems include Windows CE. The software in this project was developed for PalmOS systems, especially the Handspring Visor. More information about the Visor and about PalmOS can be found at <http://www.handspring.com>.

Communication: The geographical distribution of the team has necessitated the employment of unusual methods of communication. A decision was made not to communicate using webcams, since not all members of the team have this technology available. Two virtual teamworking software packages were investigated, 'Groove', available at www.groovenetworks.com, and 'Tango', available at www.tango.net. Both of these are centralised around a chat facility with additional sketch book sharing and screen sharing functions. 'Groove' also possesses a real-time document sharing facility, but it was found to be unreliable for the vital purpose data storage, and so 'Tango' was chosen for use, and supplemented using the 'Design Portal' website, design.engin.umich.edu, which is custom-made for this course and supports data storage and document sharing.

2. MARKET STUDY AND ANALYSIS

2.1 Project Selection

The process for selecting an appropriate project consisted of four stages:

- Familiarization with social needs and current uses of the Internet,
- Project ideas,
- Research,
- Decision.

Familiarization with social needs and current uses of the Internet: To aid selection of an appropriate Internet-ready product, much thought was given to existing social needs, which in some cases may appear far removed from the Internet but which may have viable Internet solutions. Examples are: education and skills learning, access to information, jobs and money, basic resources (food, shelter), communication, share of physical resources, entertainment, safety & crime prevention, freedom of speech and basic human rights, monitoring/caring, etc.

Additionally, we considered the current uses of the Internet, and their limitations. Uses were categorised as: e-commerce, social networks, simulation/prediction, information search, etc. Some limitations are privacy, security, speed, reliability, mobility, universality (reach), etc. A more detailed list is given in Appendix 1.

Project Ideas and Research: Based on the previous, different ideas were proposed and roughly filtered (using common sense). Some research was done to select the most suitable idea. Following is a list of these ideas and relevant research data.

- **Portable device to locate and track people or packages.** This would be used by groups of people in a busy place, to locate each other. Examples include a mother and child in a busy place, friends at a large skiing resort or music festival, etc. It could also be used to track such items as Fedex packages. It would use the GPS (Global Positioning System) technology to locate each individual to within a few feet, and the Internet to share that information with others. This idea was discarded because it would need to be connected to the Internet in remote locations where no service is available. We also found similar devices exist and are used by fleet owners to track their trucks.
- **Music sharing device.** CDs can now be copied near-perfectly and MP3's are shared worldwide, as was seen with Napster, and the music industry is suffering. The suggested product was a tool for the music industry whereby music could be sold online and written onto remote CD's for a price. This would be complemented by an Internet radio, whose stations would not only have perfect reception, but could be received worldwide either via a PC or in a simpler, specially designed, independent appliance.
- **Internet shopping device.** Internet shopping was identified as problematic, having an inherent problem of lack of physical contact with the goods to be purchased. To this end we researched a device to work with a home PC, incorporating a glove in a box of gel. The user places his hand into the box and simultaneously sees a hand appear on the computer monitor. He can then move his hand to manipulate an object (a potential purchase) on the screen, and changes in current cause the gel surrounding the glove to harden and soften,

simulating the shape and feel of the product. This could also be used for surgeons to practice remote diagnoses on patients. However, it was found that even state-of-the-art technology falls short of that needed for this project, and it was felt that far more research would be needed before a design project in this area could sensibly be undertaken. This proposal was therefore discarded.

- **Portable chat/locator device:** Based on the same technology used in the locator device, the use of this product would be to locate and chat with people that the user may know or may not know in advance. It would be a variation on an Internet messenger (ICQ, MSN, etc) that enables the user to know the geographical position of a friend and also to get to know people with similar interests by alerting him when such a person is near by. Many people chat over the Internet and may want to meet in person, and many people move to new locations in which they don't know anybody, so such a product would be useful. The technology for developing such a system is currently available and at a reasonable cost. There is a similar product in Japan that uses radio waves to match couples. The reason for discarding this idea was that we were unsure of its acceptance in different countries. Internet chat is popular because of its anonymity, and people may be reluctant to lose this privacy.
- **Kitchen companion device:** There are already some Internet ready appliances, such as microwaves, refrigerators and washing machines. Our idea was to create a specialised Internet terminal for the kitchen. It would incorporate features required by cooks, such as: recipes (either from a chef or from another user), demonstrations and shows (such as TV cooking shows, only that accessible at any time) and communication (video, voice and chat) with similar people (interests groups). This device would be capable of being placed anywhere in the kitchen. However, this solution in fact combines existing technology with little innovation involved.
- **Device to guide blind people.** Using GPS technology, this device will comprise a city map, voice recognition and audio. It will tell the user it's actual position (street, corner, nearby references) and directions to reach a specified location (address, bus stop, restaurant, etc). The Internet would be used to update maps and get new maps for other cities. A commercial GPS device can be purchased for about \$150. Voice recognition technology is also available at low costs in the form of commercial software. The only disadvantage is that is intended for a very limited market (blind people). Very similar devices were found for use in cars, so it would be a matter of adapting these for the special need of blind.
- **Device to aid informed decisions when shopping.** With so many options to choose from and so many issues involved in making a decision, the device would give the user the necessary information, which would otherwise be awkward and time consuming to obtain. It would comprise a scanner and a small screen, and the shopper would scan the barcode of a product to access information about it, which is tailored to their needs. Using either the device itself or its associated web-site, users would choose what information they required, be it dietary, medical, environmental or any other detail. Using a vast database, the device would answer these issues, and also store shopping lists, compare local prices and possibly enable consumers to buy the goods via electronic banking without visiting the checkout. No similar products were found. The technology needed for creating such a system is available a reasonable cost. A barcode scanner can be purchased for about \$100 and a wireless Internet card costs around \$100.

These seven ideas were taken to a final decision round using a Pugh Chart - containing criteria such as feasibility, market and business potential, global nature, innovation, impact on society, and skills needed. It was decided that the shopping device was the most suitable and had the most potential to attract a large market globally.

In an age of information and speed, it addresses a real need for consumers to pare down the information available to them and to make informed purchasing choices, and purchases, quickly and with ease. It allows manufacturers to publicise factual product information in a fair and unbiased forum. It allows supermarkets to offer better service to their customers and it has potential to be used by enormous proportions of society.

2.2 The Market

There are primarily three types of people who would be involved in the product: Consumers, Manufacturers and Retailers.

Consumers: This is defined as any person who makes a purchase. These could be the purchasers of the product. Research has shown that consumers have two primary needs: (1) Time saving (avoiding queues, finding products quickly) and (2) Aid in decision making (consumer reports, price comparison information, branding). In general, consumers want to be informed but don't want to worry too much about it.

Manufacturers: Manufacturers' co-operation would be needed for information on ingredients and processes. They want their products to be recommended, and to understand customers' preferences and decision making process in order to design better marketing and better products. For them, this device is a powerful tool as long as the information is real time and true. Reports could be made of what other brands the customer took into consideration before buying the manufacturer's product, and of what kind of information the consumer used to make the decision.

Retailers (Supermarkets): Retailers could be potential purchasers of the product, and their co-operation would also be needed for information on prices, etc. They want to reduce costs, to have more clients, and to be more efficient (more people, faster, less space). The device could help them to increase customer turnover and to process payments more efficiently.

2.3 Selection of Primary and Secondary Markets

Different criteria were used to select the primary and secondary countries for the launch of our products and services. Among the main ones are; local culture, people's shopping behavior, status of economic and market development, market size and technological infrastructure. Some of these criteria were useful in deciding **where** to implement the product (market) and some were useful in deciding **how** to implement it (implementation).

The primary market is chosen based on business potential, ie the need for such a system, ease of implementation, profit and growth potential, etc.

The product implementation is the global aspect. Since there are many places where the system could be implemented in a profitable way, different products (devices) will be used to implement it in different regions of the world, designed to suit different kinds of shoppers. This aspect will be explained in the following section.

The main issue is to determine where is the greatest need for such a system. The system could best be used where there is a large variety of choices and where these choices are very similar, making it difficult for the customer to determine which is the best product. It is a fact that in mature markets the above is the rule, and mature markets are found mostly in developed economies. Other important aspects in determining the appropriate markets are: market size, available infrastructure, cultural acceptance of new technologies, consumer awareness and involvement in economy.

The research was focused in countries where members of the group had lived, because of need to be familiar with the local culture due to the cultural aspect of the product. Some other countries were included where possible. Results are shown in the following table.

Country	Per Capita GDP	Product Offer (Variety)	Market Size (MM)	Internet & Electronic Infrastructure	Consumer Awareness	Acceptance of New Technologies
USA	High	High	280	High	High	High
UK	High	High	59	Medium	High	Low
Korea	Medium	Medium	48	High	Low	High
Venezuela	Low	Medium	23	Low	Very low	Medium
Japan	High	High	127	High	?	High
Australia	High	High	20	High	?	?

Using the above and the further research results found in Appendix 2, the USA was chosen as the primary region for the product, as it had the highest values for all the success factors. It also contains 165 million if the world's 426 million Internet users. The USA is an appropriate starting point for any Internet product that hopes to achieve worldwide use. Popularity in the USA would act as a recommendation to the rest of the world. The USA is also a place where, more than anywhere else, 'the consumer is king', and consumers not only have a wealth of conflicting information, but also real power to affect prices, ingredients, production methods etc of the goods they buy. Adding to this, the 'melting pot' culture of the USA, whose society is made up of many different ethnic, cultural and religious groups, mean that many consumers have vastly divergent concerns dictating the products they buy. Consumers in the USA are therefore accustomed to making responsible choices, and by organization of the information available to consumers, the USA is the place where our product can have the largest effect and be the most easily accepted and integrated into common use.

Another region in which consumers already attempt to make responsible shopping choices, although bombarded with conflicting information, is the UK. Language and culture similarities between these two countries mean that products often cross the Atlantic smoothly between the two regions. The UK also shows the 2nd best values for the main research points. Therefore the UK was chosen as the secondary marketing region.

3. 'GLOBALITY' AND ADAPTABILITY – JUSTIFICATION FOR MARKET AND PRODUCT CHOICES

'Globality': The product relates to shopping, specifically for groceries although in the future it could apply to any kind of shopping. Shopping and the power of the consumer are features of every civilized society, and as information becomes more readily available, consumers become ever more discerning in their purchasing habits. This is a clear global trend, and although it exists to differing degrees in different regions, all civilized societies are moving towards an age where a product's healthiness, ingredients and manufacture have at least as much influence on consumer choices as its taste or function. In short, all industrialized countries will soon have a use for this product.

The device will identify the product using its barcode. Global standards already exist for barcodes, and so the scanning technology will not have to be changed between regions, thus making the product highly suitable for global use.

In any large modern supermarket, imported goods can be found. Goods made in the USA, for example, are commonly sold in the UK and all over Europe. Using an Internet based database and this global information device will mean that information does not have to be duplicated for each country. A shopper buying an American food product in France will simply access the same data as an American shopper buying the product domestically.

Furthermore, as corporations become global, their policies and operations affect many countries in terms of such factors as pollution, employment, etc. Thus it is an industry that lends itself to globalization.

Adaptability: The core part of the system, which is the information, will be common to every country (an universal products database) and there will be different ways to access this information in different countries, adapted to the specific needs of each country. Thus family of products is needed. This is shown in the hierarchy in section 4. A different blend of these product variations, along with specific strategies for selling, marketing, minor aesthetic and language changes will be needed for each different country.

Research was conducted, based on a survey and on observation, to determine a product mix for different countries. The sample survey is given in Appendix 3 and survey results in Appendix 4.

Research - Concerns: A survey was conducted in several Regions: USA, UK, Korea and Latin America. The results showed different consumer concerns and attitudes about food and groceries shopping. These have been summarised in graphs in Appendix 5.

In Korea, the biggest concern is about the freshness and the quality of the food. In Latin America, supply seems to be a problem and so is the time spent at supermarkets, especially at checkout. Other issues are the reliability of the stores and problems locating the products within the supermarket. In the UK, the time spent at checkouts is the biggest concern, followed by supply, service and locating problems. Finally, in the USA, customers worry the most about checkout time and location, but also claim for more information and individualisation.

From this part of the survey, we can see some improvements that we can fulfil through our system. These include time at checkout, locating products, improvement of services and quality (by real time suggestions and surveys), information and individualisation. Issues that

are out of our scope for the moment include freshness, reliability of stores and supply. This leaves us with just two regions in which our system can be better used by the majority of the consumers: USA and UK, reinforcing our previous decision to work in these 2 countries.

Research – Attitudes: Another part of the survey consisted of a series of statements, which implied several attitudes of the consumer. Participants expressed their agreement or disagreement with the statement, from 1 to 5, 1 meaning a complete agreement and 5 meaning a complete disagreement, 3 a neutral point. Each answer implied several attitudes through shopping, which were processed and classified into different categories. The results from this part show the different attitudes of people from the different regions, reassuring that SIS's products can be used world-wide for different needs. The results are displayed in Appendix 6. These results were used as a reference and not as a conclusive study due to the small size of respondents.

Research - Observation: Supermarkets of different sizes in various regions were visited and observations were made. It was found that shoppers behave differently from each other. They were grouped into four categories, as follows.





- **In and Out Shopper:** This shopper wants shopping trips to be very fast, knows what s/he wants, is not distracted by sale signs, walks quickly with determination - like a New Yorker. S/he never spends very long making a decision, looks at a sea of products and just picks one, usually buys familiar products to speed up decision time and get out quickly, and often doesn't use a cart. This type of user would have more tendency to own his own device, or be able to use an individual device, already pre-set to his preferences, mainly because of time restrictions.
- **Intuitive Shopper:** This shopper often has kids in the cart, but not always. S/he navigates through almost every aisle and looks for products that "catch-the-eye". Often s/he will pick up several products to compare them -looking at all sides of the package, but not really reading them. S/he seems to make decisions based on the graphic design of the package and some intuitive feeling for the product's quality, etc. This kind of shopper is likely to use also an individual device, mainly because of comfort.
- **Informed Shopper:** This shopper is more rare - often someone with very specific needs, like a vegetarian, allergy sufferer or orthodox Jew. S/he spends time reading packages to help make decisions about which product to buy and can explain their decision-making process rationally if you ask them. As this user normally spends more time at the shop, s/he is more likely to use a public terminal of the system.
- **Small Town Shopper:** This shopper visits small, local stores regularly. S/he knows the clerks at the store, and they know her/him. Shopping is more social - not as task-oriented. Also this might apply to street-market shopping in China, India, etc where shoppers haggle prices and discuss products with sellers (often arguing). This type of shopper is more rare in USA - only in small towns. In US, we almost never talk to the workers at the store - and if we do, it's just to ask "where's the bread?" The best product for this user will be a store owned clerk operated terminal, so the user still interacts with a person, rather than with a computer.

See Appendix 7 for observation results in the different regions. In conversations with the managers of the supermarkets and in general they agreed in this classification, though they cannot quantify them.

With the above target customers in mind, the SIS product family was created. Different products, all using SIS information services, would meet specific target users needs:

- **Aware / Easishop:** For commercial markets where PDA's are widespread and popular, where people like to own their own devices, customized to themselves, and where info privacy is an issue. Good for North America, Japan, Hong Kong and similar countries.
- **BuySmart:** For markets where PDA's are not widespread, but people like to own their own devices - technology friendly markets. Good for Europe probably, and maybe Canada.
- **SuperScan:** For markets where consumers are not willing to invest in the product, but stores may be willing to, in order to differentiate themselves from the competition. Can be used to attract people to try-it-out by advertising coupons and sales for using it. Good for Central and South America and for some of Asian countries like Korea. It can also serve as a good pilot for the INFO3000.
- **INFO3000:** For markets where consumers are not willing to invest in the product, but stores may be willing to, in order to differentiate themselves from the competition. Probably used in upper-class stores. More efficient than SuperScan (no waiting in line to use it), but costs more for the store, and harder to manage and maintain. Maybe SuperScan can be used as a pilot study, and if people really like it, stores can switch to INFO3000. Probably best for North American markets.
- **Megascan:** For markets where consumers are generally not computer literate and not interested in trying high-tech devices. Works in markets where consumers tend to talk to store workers, and also good for small stores rather than supermarkets. Good for India, Egypt, and some South American countries.

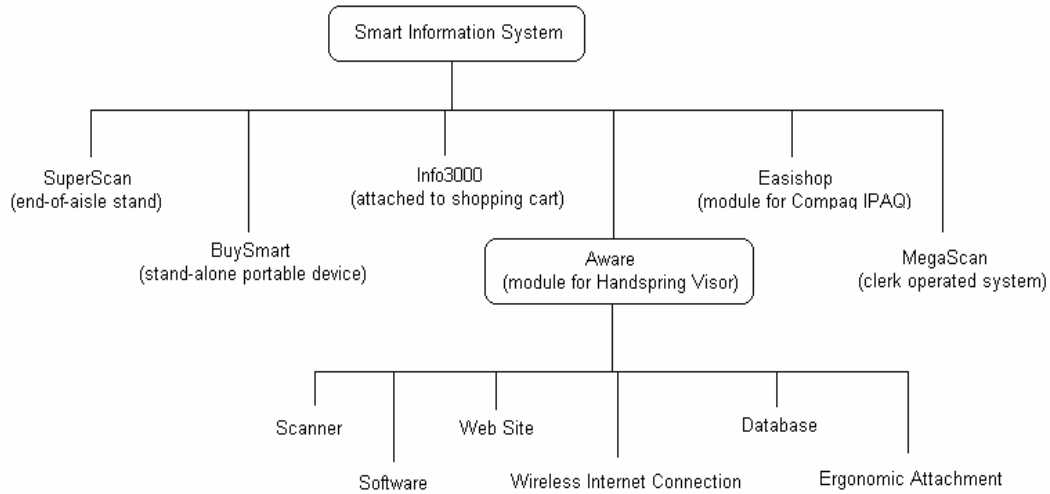
To summarise the above, the following matrix was constructed taking into account the results of the above studies.

Country	Type of Shopper				Type of Device to Offer			
	In and Out	Instinctive	Informed	Small-Town	Personally-Owned PDA Module	Store-Owned Cart-Mounted PDAs	Store-Owned Walk-Up Touch-Screen	Store-Owned Clerk-Controlled Terminals
 Korea	10%	60%	25%	5%				
 United Kingdom	40%	30%	30%	0%				
 United States	55%	30%	10%	5%				
 Venezuela	35%	50%	5%	10%				

This illustrates how the research described above points to the use of a central information system with a variety of outlets or ‘sub-products’ tailored to the needs of different regions. The green areas are the ones with the highest potential use for the product, hence these are the chosen primary and secondary regions.

4. PRODUCT DEVELOPMENT FOR THE USA

The Sub-Functions: The Smart Information System and its functionality are best explained by use of the hierarchical diagram below. The first level shows the sub-products of the Smart Information System, and the second level shows the sub-functions of the one sub-product chosen for development (Aware™) for the primary market.



To ensure that functional and aesthetic prototypes could be delivered within the given time frame, it was necessary to choose one product for detailed development out of all the sub-products of the SIS. This product was Aware™, chosen since it is most suitable for the primary market (the US) for the reasons outlined in the section on Global Use and Adaptability, and because it makes expedient use of Team Five's skill base, as detailed below. The sub-systems defined in the diagram are described below:

No.	Sub-Function	Description
1	Scanner Module	Barcode scanner attached to PDA.
2	PDA Software	Software to provide the interface between the user, scanner and database and the display screen on the PDA.
3	Web Site	Alternative means of accessing user profile etc.
4	Wireless Internet Connection	WLAN card in the PDA.
5	Database	Structure of central information store and access to the information.
6	Ergonomic Attachment	Clip or pocket to hold the device on the shopping trolley (shopping cart) and make it easy to use.

The Tasks: For the purposes of development, it was necessary to split the sub-functions further into tasks and choose areas to analyse in detail. Considering that the wireless Internet and scanner can be purchased ‘off-the-shelf’ as modules for the PDA, it was decided that in order to produce a working prototype, the best plan was to develop three of the sub-functions in detail: the PDA software, the database and the ergonomic attachment. The latter also required consideration of the size and shape of the Aware™ module itself. The choice of these areas for development takes advantage of the CAD and programming bias of Team Five’s skills, and will provide tasks appropriate to the given time scale and equipment. The web site would in fact be very similar to the PDA interface, perhaps using more graphics, and so this sub-function will be taken no further as it would duplicate work to do so. The scanner and wireless Internet sub-functions, however, would benefit from some attention to adapt them to the product’s use.

Thus tasks for further development have been selected as follows:

No.	Sub-Function	Task(s)
1	Scanner Module	Research into and adapt relevant technologies.
2	PDA Software	a) Learn coding language, b) Develop requirements of Aware™ software, c) Design interface, d) Programme PDA to demonstrate some functionality.
3	Web Site	-
4	Wireless Internet Connection	Research into and adapt relevant technologies.
5	Database	a) Research structures of distributed databases, b) Develop plans for architecture of database, c) Make prototype database with sample records and partial functionality.
6	Ergonomic Attachment and Module Design	a) Research shape of shopping trolleys and ergonomics of use, b) Design attachment and module in CAD, c) Make functional and aesthetic attachment and module prototypes.

4.1 Implementation

The above sub-systems must synthesise to form a single product whose separate parts work in unison. The following sections describe the sub-systems designed by Team Five in detail.

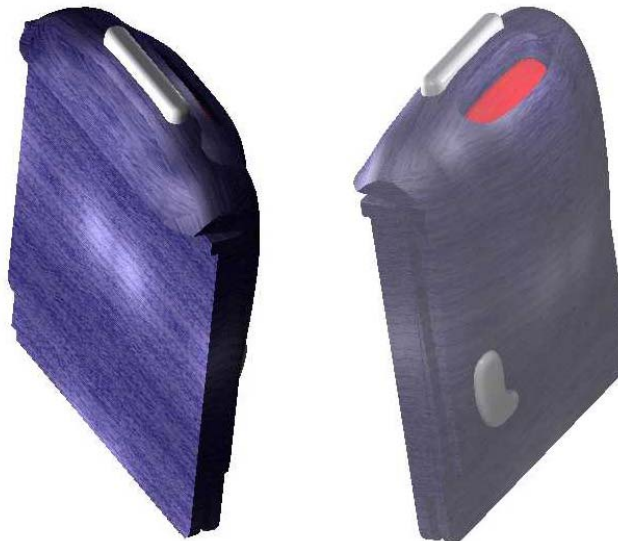
4.1.1 Scanner Module & Wireless Internet Connection

The Design: The designed module combines a Symbol scanner unit and a WLAN card, and it is designed to connect to a Handspring Visor expansion slot. The internal electronic components have not been designed, but instead a case has been designed that is large enough

to house the major components and fit correctly with the Visor. Handspring provides extensive documentation about standards for their expansion slot, and our module is designed to meet the requirements. The module has a large 'Scan' button at the top that enables users to turn the scanner on and scan with one hand by simply pushing it against the barcode. It also has a 'Buy' button on the front.

The Functional Prototype: For demonstration purposes, we purchased a Symbol CSM-150 module for the Handspring Visor, which performs scanner functions but does not include a WLAN. Instead, we simulate the function of the WLAN and central database by programming a small database directly onto the Handspring PDA. Both our design and the Symbol CSM-150 module will work with any Handspring PDA, and the Aware™ software will work with any Handspring that uses PalmOS 3.0 or above. Almost all PDA devices use PalmOS 3.0 or above.

The Aesthetic Prototype: The module was drawn using Solid Works CAD programme and manufactured in a nylon material using rapid prototyping. It was then sanded to fit into the Handspring and painted using spray paint to indicate its aesthetic. The CAD drawings of it are shown below.



Wireless Internet: Although the software designed for the prototype includes a small database, in the final design, the PDA will not have a database installed on it. Instead, a central database server is accessed remotely from the PDA via WLAN (Wireless Local Area Network) This is dealt with in detail later. When the user scans a product, the PDA sends a request through the store's WLAN to a central database HTTP server. The server manages requests, and it responds to queries by sending appropriate information. The PDA then displays this information according to the preset user profile. For instance, a user may preset an alert to the presence of nuts. The PDA will therefore query the database about the nut content of each scanned product. The database server will respond, and if the item contains nuts, then an alert will be triggered on the product information screen of the Aware™ interface. This process, from scan to screen, will take under one second.

4.1.2 PDA Software

The Aware™ software is a graphical user interface used on a PalmOS PDA, such as a Palm Pilot, Handspring or Compaq. The software controls the scanner module and WLAN card, and it formulates query requests for the central and in-store databases. The software was developed using a Software Development Kit (SDK – a library of functions) provided by Palm, using the C programming language in a Metroworks CodeWarrior environment. A demo version of the CodeWarrior compiler was used (to avoid a \$500 cost for the full version), so it was necessary to keep the application under 5K in size. This limitation prevented all of the functionality from being implemented in the prototype, but the major functions are there in order to explain the idea. The sources for the development software documentation are given in the reference section.

The major functions of Aware™ are as follows.

Scan: Scans a product barcode, finds the product in the database, and shows that product information on the Product Information Screen.

Information: If the barcode is not available, product information can still be looked up, by selecting the product from the full list. If the barcode is available, and it is scanned, then the Product Information Screen is accessed with the scanned product automatically selected. From the information screen, the user is able to buy the product if they have an electronic credit account set up, and if the store is set up for consumer self purchasing. Security may be an issue that makes consumer self-purchasing difficult; however, several solutions may make this feasible, including the warehouse-style store (see Cart below).

Compare: Scans two products and shows their information side-by-side. The same screen is used when the customer uses the “Suggest Alternative Product” function. In this case, an alternative product is found that best matches the user criteria, and the product information is shown next to the original product. This function also enables a user to scan one product and receive information about all similar products in the shop (store) which s/he is in.

Preferences: The preference screen allows the user to set preferences about what kind of product information s/he wants to be alerted about. All product information is always available through the database. However, information that is set up for alerts will be brought to the user’s attention. The preference screen allows the user to set up alerts about

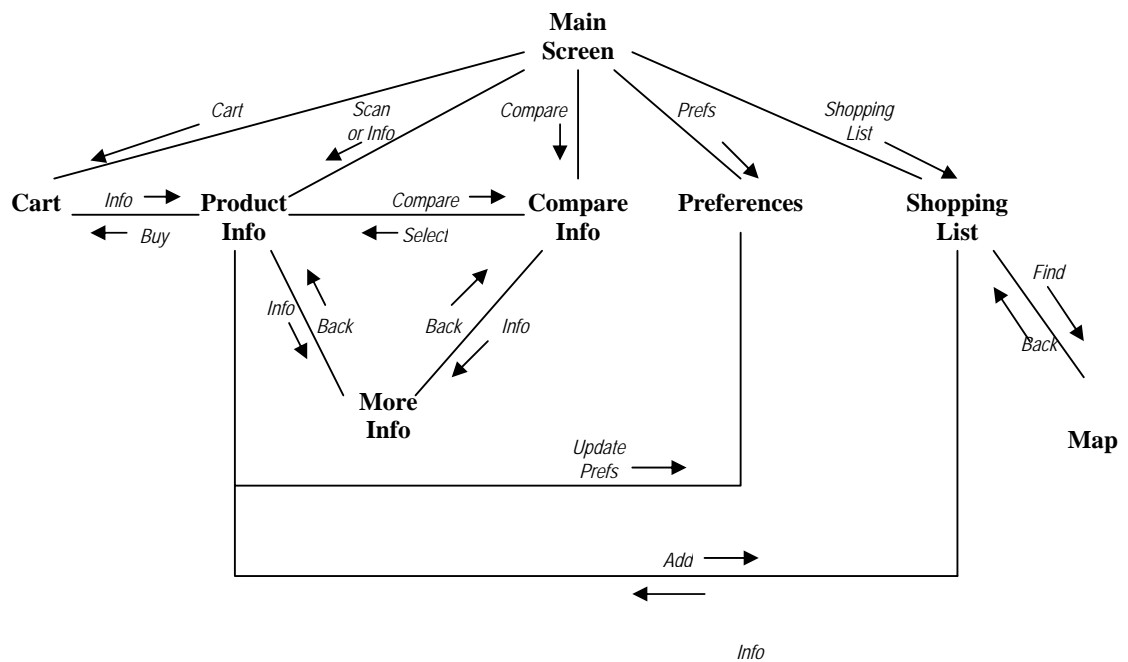
- product ingredients,
- nutritional information,
- processes used in producing the product (such as animal testing),
- corporate policies (such as environmental policies),
- endorsements (from groups like Greenpeace or Consumer Reports),
- preparation and maintenance information (such as need for refrigeration after opening),
- packaging (materials and inks used, recyclability, etc)
- price.

Shopping List: The shopping list screen allows the user to keep an electronic shopping list of needed items. The list can be added to manually (by writing items in), or automatically (by scanning used containers).

Cart: The cart shows a list of items that have been purchased, and it can calculate the total cost. This feature is especially useful for warehouse-style stores that may emerge from extensive use of Aware™. In the warehouse-store concept, customers shop by scanning products on shelves; however, they do not physically take the items off of the shelves. Instead, as the item is scanned, workers in the back of the store are immediately notified through the WLAN, and they begin collecting objects from the back-of-store warehouse. When the consumer is finished, the workers can bring the products directly to the consumer’s vehicle. This system reduces the need for restocking and price updating, and it may allow reduction in the size of the store because the system only requires one of each product to be available to the consumer.

Suggestions: This option allows shoppers to send quick suggestion notes to the store or the product manufacturer at the time of shopping. If something bothers a consumer about the product, this system will provide an immediate mechanism for feedback that allows the consumer to avoid making a fuss in person and holding up lines or tracking down managers. This feedback mechanism is very valuable to stores and manufacturers because it will enable them to make more informed decisions about how to meet customer needs. In fact, this is the main feature of Aware™ – it enables every member of the shopping experience to make more informed decisions.

The diagram below shows how the menu system is structured. Most of the menus were implemented for the prototype; however, not all could be implemented because of code size limitations with the compiler. In the diagram, bold face items refer to screen titles (called form titles in PalmOS terminology), and italic items refer to button presses. In addition to the links shown, each screen contains a “home” button that returns the user directly to the Main Screen.



Sample screen shots of the Aware™ program are shown below. The first screen is the home page, the second is a user profile and the third gives information on a scanned product.



The code used to programme this software is given in Appendix 8.

4.1.3 Smart Information System: Database Structure

The database is the essence of the Smart Information System, and through it all the above-mentioned products access information over WLAN. Since it must include information about hundreds of thousands of products in each country, and since it will need to be constantly updated, it will have a fairly complex structure and deserves some attention here.

When a user sends a query from their handheld Aware™ device, the information will follow a specific path to the source of the information and back again. This path is illustrated below, to show the global picture. The following sections gradually break this path down into its component parts to give a full and detailed description of the architecture of the database.

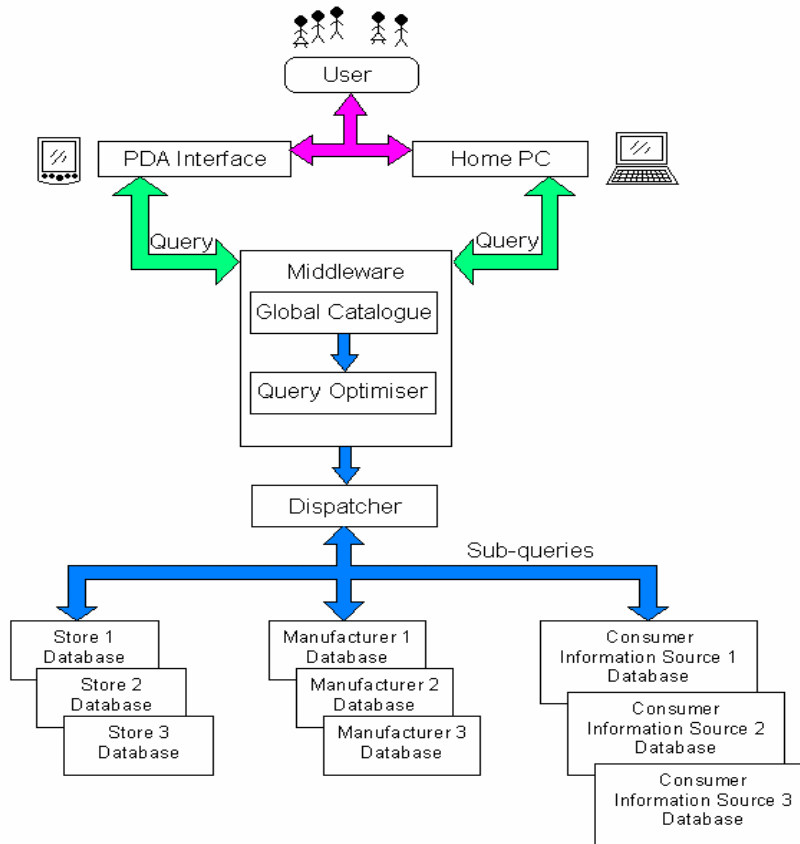
The information flow chart above can be explained as follows. When a user queries a product or changes their user profile, the information flows from the user to the PC or Aware™ device being used.

Information is then passed to the global manager or middleware, which decides what to do with the information. User profile information is stored, and queries are passed to the query optimiser. It should be noted that many devices may access the same global manager at once.

The query optimiser, among other tasks, decides which external database contains the information required by the query and then requests it (via the dispatcher) from the appropriate external database. These external databases come from a variety of sources and are discussed in detail later. The information is then sent back to the global manager and from there to the PDA and to the user.

Distributed Database: As the Information Flow Diagram indicates, the data comes from many external databases. This implies that there are many independent servers, and thus the system has the characteristics of a distributed database. The distributed database structure was

developed as a pragmatic response to the evolution of geographically distributed organisations, and as such it is very well suited for use here.



Middleware: Since the functions of client and server overlap somewhat in this particular system (due to the complex relationship between queries and servers), a middleware system will be used. This will handle single queries spanning multiple servers without the need for each server to be able to address the other independent servers. It will not be possible to alter the configuration of the external databases, and so the middleware will accommodate this. The middleware is essentially a layer of software that co-ordinates execution of queries and transactions across one or more independent database servers. It contains a catalogue, which tells it what to do in the case of system mismatches, and an optimiser, which plans the best way to answer a query. Using the middleware also means that the databases are effectively unaware of each other. This is necessary, since the supermarkets and manufacturers are unlikely to want to share information.

Thus, the individual client sees a regular DBMS (Database Management System). User profile data is stored in the middleware, but all other data is stored in the external databases 'behind the scenes'. Queries can then span many external databases (ie many servers), and the middleware uses its knowledge of the capabilities of the external systems to formulate globally optimal query plans. Query optimisation is dealt with in Appendix 9, as well as the specifics of enabling co-operation between independent databases.

Advantages: The distributed database structure allows data and control to be distributed among the environments in which it is generated and used. This allows the use of pre-existing databases belonging to the stores and manufacturers, which will be necessary here and is discussed below in detail. A distributed system allows greater flexibility and failure

resistance, and also allows much easier progressive addition and modification than databases based on central mainframes, which will be very important here as prices and ingredients are constantly changing.

Classification: The system will use WAN (wide area network), and will by necessity be heterogeneous, ie will use more than one type of DBMS, since it will interact with many external databases. This implies a need for an open architecture to ensure compatibility of the databases, which is discussed later.

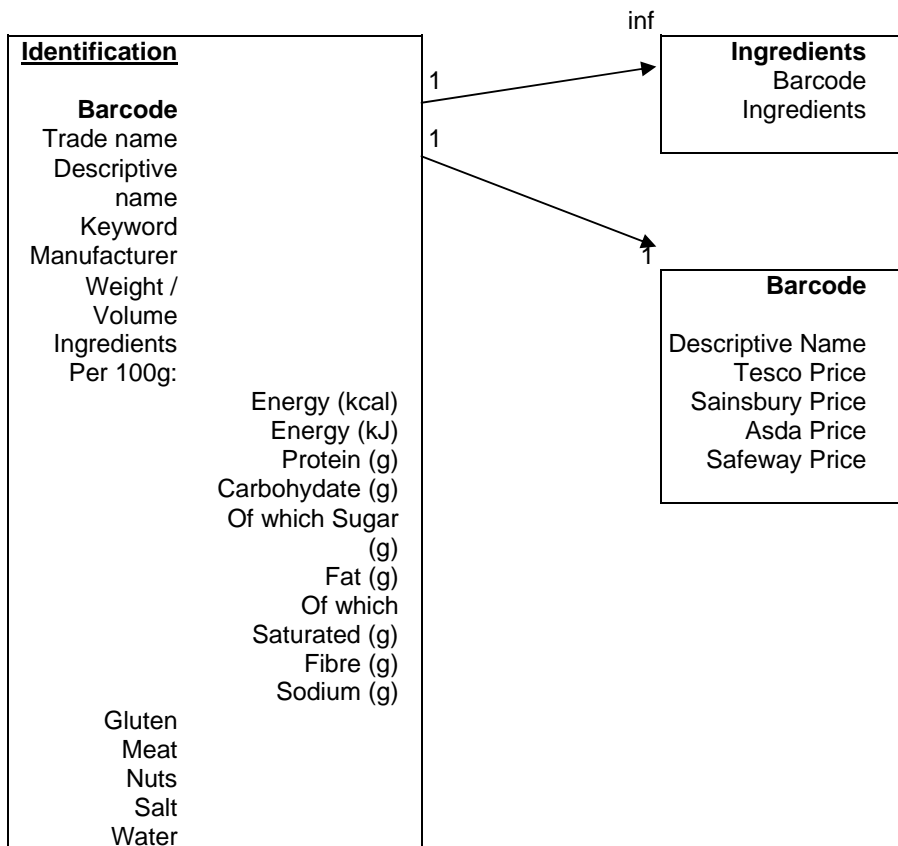
Local Independence: Much of the data will already be distributed (in the external databases) but where possible, data distribution should be planned in such a way that as many applications as possible can operate independently on a single server. For example, user profiles should be stored on one server. This avoids the large execution costs that are typical of this type of system.

Transparency: The system will be characterised by 'allocation transparency'. All servers can be queried using the same language. The Global Management Database 'knows' the fragmentation of the databases but does not have to indicate the allocation of fragments. In particular, to answer a query about ingredients in Hovis Bread, the information may be replicated in several store databases, but there is no need to specify in the query which replica is required. This additional property is 'replication transparency'.

4.1.3.1 Database Implementation

Information Acquisition: The system outlined above relies heavily on the willingness of the external database owners to share their data. This has been investigated in some detail. Research was done to ascertain the largest supermarket chains, manufacturers and consumer information sources in the proposed markets and a letter or email was sent to as many of them as could be contacted, detailing the proposed product and asking for their opinion on the likelihood of their compliance. Both the letter and a list of contacts are included in Appendices 10 and 11 respectively, along with a sample response in Appendix 12, and there was enough interest in the project to indicate that it is likely to be possible to obtain compliance from such sources of information.

Prototyping: A prototype database has been made. It contains details about the ingredients and source of 28 products, and their price in the four largest UK supermarkets. This information was gleaned from packaging and from supermarket web-sites. The database is written on Microsoft Access 2000 since this is adequate for a small prototype, although a more complex programme such as Microsoft SQL would be needed for the real one. It is a relational database with tables whose structure and contents are shown in the diagram below.



The prototype database contains three pre-set user profiles and answers two queries:

- **Single Scan.** A user scans a product's barcode and the database returns the information specified in their user profile about that product.
- **Compare.** A user scans a product's barcode and the database returns the information specified in their user profile about all products similar to the one scanned.

The queries were written directly in SQL and the code is given in Appendix 13.

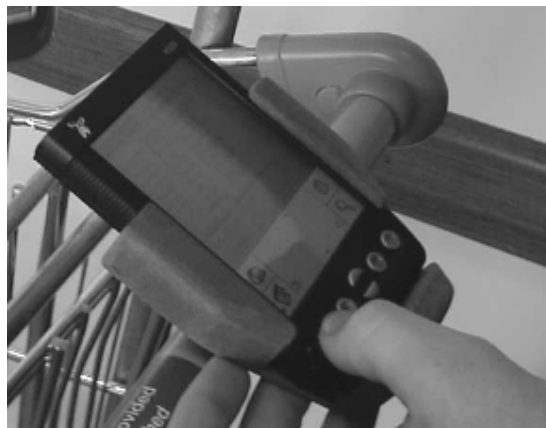
4.1.4 Ergonomic Attachment

The dimensions of the Handspring and attached module have been used to design an ergonomic attachment. A CAD drawing of this is also included here. The attachment has been designed to fit onto a shopping trolley. Thus, the scanner is attached to the trolley, enabling easy scanning of products simply by holding them above the PDA and nudging the 'Scan' button on the module.

Virtual Prototype: The size and shape of the Handspring Visor and the module have been designed and illustrated in CAD drawings, shown below and on the front cover.



Physical Prototype: In addition, a physical prototype has been made from plexiglass using a milling tool. This was sanded and spray painted and attached to a supermarket shopping trolley for demonstration purposes. It is similar in design to the virtual prototype but priority was given to ease of manufacture here since this was a one-off production. It is shown in use in the photograph below.



4.2 Production

Electronic parts for the manufacturing of the Aware™ module are already being manufactured for many other applications. The only new part to manufacture is the enclosure, which will be made in injection moulded plastic.

As the main manufacturing activity will be assembling, not requiring highly skilled labour, it will be done either in Taiwan or China. All parts, which are probably manufactured in those countries, will be shipped, assembled and the Aware™ modules shipped to the United States.

The estimated cost of an Aware™ module in the USA is \$105. A table of costs is given below.

description	6 cavity injection mould approx cost	units per year	cost per unit
mould 1	40000	1000000	0.04
mould 2	40000	1000000	0.04
	cost per year	units per year	cost per unit
material 1	200000	1000000	0.20
material 2	200000	1000000	0.20
labour 1	40000	1000000	0.04
labour 2	40000	1000000	0.04
assembly labour	2400000	1000000	2.40
	cost per stroke	units per stroke	cost per unit
machine use	5	6	0.83
machine use	5	6	0.83
manufacturing costs			4.63
components cost	100		100
cost at factory	shipping cost per load	units per load	
transport	4000	83333.33	0.048
cost in USA			\$ 104.67

It is estimated that it will be necessary to charge a service fee to cover for the development and maintenance of the information database and server, although that issue was not addressed due to time limitations.

5. PLANS FOR LAUNCH IN THE UK

The Product: As explained above, the secondary market for the Smart Information System is the UK. The sub-product of the SIS to be launched there is the Info3000™. This is the store-owned adaptation of the system, and as such it is intended to give UK customers almost identical functionality to the Aware™, with far smaller outlay.

Adjustments and Constants: The Info3000™ is contained in a small unit which is approximately the same shape as the Handspring Visor, and which remains permanently attached to the shopping trolley. The unit contains the same wireless Internet and scanner modules as Aware™ and the software is such that its screen shows exactly the same interface as the Aware™. It accesses the same Smart Information System database; the same user profile storage facilities will be used as in the USA, and some of the same data will be used for products which are sold in both countries. The database will be extended to include information from British manufacturers and supermarkets, but its structure will remain the same.

To retain the essential 'personalisability' of the Smart Information System, the product will be launched in conjunction with a web site. The web site will navigate in the same way as the Info3000™ software, and users will set a user name and password in order to access and edit their user profiles and make shopping lists, from their home PC. These functions will also be available on the Info3000™ itself, but it is expected that new users will want to familiarise themselves with the system in private.

For a smaller price than that of a PDA, large supermarkets will therefore be able to buy many Info3000™ units to attach to their shopping trolleys. The difference in price is largely due to the very limited software that these units will contain. To reduce the price further, it may be possible to use larger (therefore cheaper) scanner and/or wireless Internet equipment, since the Info3000™ will never be carried in a pocket or handbag (purse). This would require further research.

The units will need to be securely fixed to the trolleys so that they cannot be removed. They must be made from a resilient material with reinforced screens so that they cannot be vandalised, and to minimise wear and tear. Shopping trolleys are often taken outside to the car park (parking lot), therefore the unit must be fully waterproof. It is likely that supermarkets will place a higher priority on car park security and trolley collecting as a result of this product, but that is at the discretion of the individual supermarket.

Market Launch: The launch is planned for one year after that of the Aware™ in the USA, to allow time for any defects in the software to be altered, and time for any relevant feedback from the USA to be made use of. The beauty of the British version of the product is that the public will not need time to familiarise themselves with the product in advance of the launch. The product will be available to them straight away for a free, commitment-free trial at their local supermarket. The parties that will need to be convinced, however, are the British supermarkets. Feedback from these as detailed in section 4.1.3 (Database Implementation) and in Appendix 11 indicates that at least one major chain would be interested, as they are 'always looking for ways to differentiate themselves'. The other three large chains would then be very likely to follow suit in an effort to compete.

Thus it is expected that the Smart Information System will successfully penetrate the UK market in the Info3000™ incarnation.

6. CONCLUSIONS

Objectives: The brief was to design a product for Interea Inc which was both global and Internet ready. The global aspect will be dealt with first, with reference to the definition of ‘global’ in the introduction. The Smart Information System and its outlets or sub-products have been designed taking into account detailed market research in the UK, USA, Korea and Latin America, as well as some in other areas of the world. The information itself would remain constant throughout the regions of the world while different outlets would be used in different regions. One of these, AwareTM, has been designed in great detail for use in the USA, and justification has been given as to why it is appropriate here. A second sub-product, the Info3000, has been described briefly and is to be used in the UK. Justification has been given for this also. Thus, the Smart Information System is truly global.

The other objective was to make the product Internet-ready. This was defined as a product ‘that not only makes use of the Internet, but answers a real need that is present in society, a need that is best answered by using the Internet’. The product certainly makes use of the Internet, and in fact, there would be no other way of implementing such a system. The information must be constantly updated, with disastrous results if this is not the case. The use of the Internet and databases from supermarkets and manufacturers ensures that information will always and automatically be current.

Lessons: Aside from the brief given, the project had an intrinsic aim of fostering global relations and of teaching virtual teamwork. Communication and time differences were huge problems initially and much time was wasted in familiarizing ourselves with software and duplicating work. Team Five would advise future teams to use the communication software which we chose in section 1 (Introduction), and to use the time in the first week to allocate discrete tasks to each team member to ensure that work is done even when communication breaks down. It is also worth noting that while words can often be misinterpreted by a non-native speaker of the language, pictures, diagrams and flow charts are far more universal.

At the end of the project the team had adapted to each others’ styles of communication and, to some extent, had overcome language barriers and found ways to understand each other. Global, virtual team-working is certainly a skill that can only come from experience, and since it is becoming increasingly necessary in today’s professional world, it is hoped that our experience can be put to good use.

References:

Market Research:

City Population: <http://www.citypopulation.de/>

World Comparison of Real Gross Domestic Product and Purchasing Power:

http://www.un.org/Depts/unsd/sna/gdp/gdp01_hm.htm

University of Texas Library Online:

http://www.lib.utexas.edu/maps/world_maps/world_gdp96.jpg

Hardware Used:

Handspring Visor: <http://www.handspring.com>

Symbol CSM-150 <http://www.symbol.com>

PDA Software Development Tools:

Palm SDK 3.5: <http://www.palm.com/developers/>

Handspring Libraries: <http://www.handspring.com/developers/>

Symbol CSM-150 SDK: http://www.symbol.com/services/downloads/download_mobile_spt1500.html

CodeWarrior for PalmOS Version 8.0 Evaluation:

<http://www.metrowerks.com/products/palm/>

Database:

Ramakrishnan - *Database Management Systems*, 2nd Edition, McGraw Hill (2000)

Connolly – *Database Systems*, 3rd Edition, Addison Wesley (2001)

Atzeni – *Database Systems*, McGraw Hill (1999)

Buyens – *Web Database Development, Step by Step*, Microsoft (2000)

Habraken – *The Complete Idiot's Guide to Microsoft Access 2000*, MacMillan (1999)

APPENDICES

APPENDIX 1 – USES AND LIMITATIONS OF THE INTERNET

Uses

1. E-commerce
 - Money transactions
 - Music
 - Movies
 - Entertainment
 - Any goods
2. Social Networks
 - Sports
 - Game playing
 - Meeting people
 - Virtual communities
 - Email
 - Chat rooms
 - Communication
3. Simulation/prediction
 - Distributed computation
4. Information
 - Research
 - Organisation
 - Control
 - Manipulation/monitoring of artefacts

Existing Social Needs and How the Internet Could Help

- Skills – via information, online courses, etc, sharing knowledge
- Access to information – via technology (search engines)
- Jobs & Money – via new jobs created in e-commerce, via Internet job hunts
- Basic Resources – via knowing where to get products, price comparisons
- Communication – cheap long distance email and phone
- Sharing physical resources – tools, appliances etc – via Internet
- Entertainment – see above
- Safety & Crime Prevention – via monitoring, control
- Political boundaries – via access to information, freedom of speech
- Monitoring/caring for things while away, eg pet, plants, home – via camera and appliances
- Etc.

APPENDIX 2 – INITIAL REGIONAL RESEARCH

USA:

Special Interests Groups:

Diet restricted: Interested in nutritional and ingredient information. Diabetics. (16 MM in US), High blood pressure. (50 MM in US. + 2 MM each year in US), Kidney disease, Obesity. US: 22% (55 MM in 1990), Kosher, Halal, Vegetarian. 20% of US population (not strictly vegetarian), Cholesterol, calories, diets. Lots of people!!!

Environmentalists groups: Their power relies on spreading their ideals (public opinion). Their main effort is in campaigns to attract the attention of the media and create public awareness: **GreenPeace** 2.9 MM supporters in 158 countries. \$ 90 MM 1999 revenue, 169.000 members in UK, 250.000 members in USA. Sub groups: Toxics (Industry), Forests, Oceans (fishing), True food network. 20.000 members. Against genetically engineered food, features: True Food Shopping List. **Co-op America** presents thousands of socially and environmentally responsible products and services. Features: Product finder, Product guides. **GreenMarketplace.com:** (Private Company). Sells Green Products. Want to reduce toxins, pollution, waste, and cruelty. The Campaign.org (to label GM food)

Pro Animal Rights: People for the ethical treatment of Animals (PETA). Animals are not ours to eat, wear, experiment on, or use for entertainment. More than 700.000 members. (in USA), Features: List of companies that still test in animals.

The above demonstrates the big concern for different issues regarding to products, which our device Should Address.

Product reviews. Aid in decision making, test products and make comparisons and recommendations. E.g. Consumer reports Magazine. 4.000.000 Readers, new-new.com/ (Europe), reviewboard.com/, savvyconsumers.com/, which.net/ UK. This demonstrates that efforts addressing this need are already wide spread trough the U.S.and Europe.

Korea:

Korean Consumer Attitudes

Summary of 1999 survey of Korean's consumer's attitudes on genetically modified foods (GMF)

- Do you think that you're probably eating GMF?
 - 51.0 % DON'T KNOW
- Would you eat GMF when you get to know it's genetically modified?
 - 56.7 % IN SOME CASES
 - 21.9 % NEVER
- Is it necessary to add some special notice on the GMF?
 - 71.5 % NECESSARY

Summary of 1999 survey of Korean's consumer's attitudes on Food Labels

- How often do consumers check information on the label when they buy foods?
 - 46.2 % Often
- Why do consumers check the label?
 - 94.4 % Expiration date
 - 40.7 % Country made in
 - 39.0 % Ingredients
- Are consumers satisfied with the label?
 - No. People want the information to be expressed in DIAGRAMS.

Korean Consumer Concerns

1) Consumer's role

Consumer should ask for precise information of a product. (90.6 %)

2) Consumers' thought about safety of food/pharmacy

Quite large difference of safety between domestic and imported products;

think that domestic products are safe, but imported ones are not safe

Beef/pork; domestic (3.66 pt of 5), imported (2.23)

Vegetables/fruits; domestic (3.51), imported (2.29)

Beauty products and pharmacy are generally safe.

Tap water, frozen/processed foods are not safe.

3) Generally not quite concerned to do something for environmental problems.

4) Most serious problem

Consumer safety problems resulted from bad/harmful foods (27.7 %)

Problems like false or exaggerated advertisements (26.7 %)

APPENDIX 3 – CONSUMER SURVEY

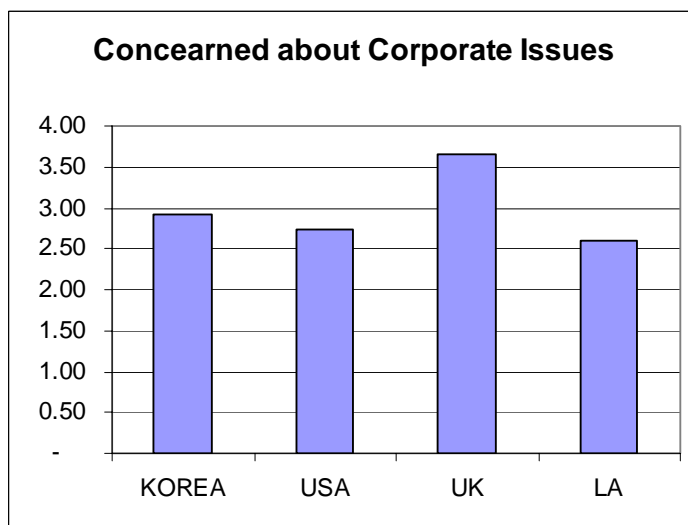
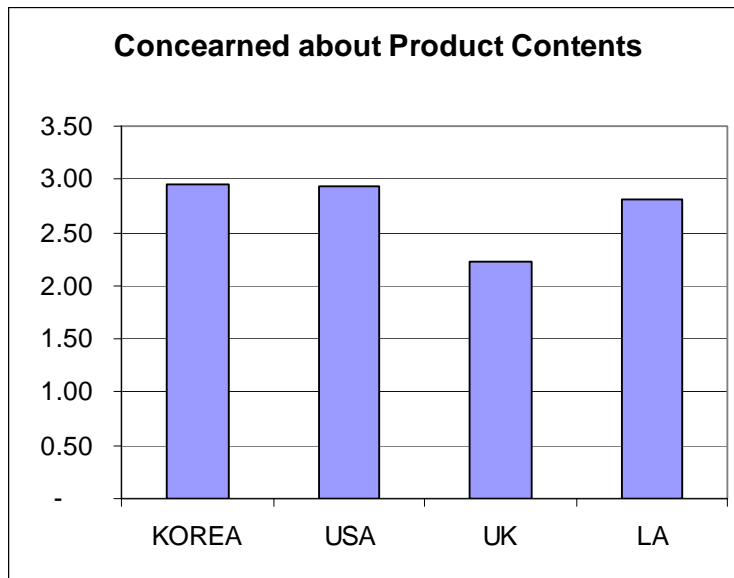
1. Basic information:
 - a. Age _____
 - b. Sex M__ F__
 - c. Married Y__ N__
 - d. # of children _____
 - e. Country of residence _____
 - f. Total Household Income
 - a) Below 10.000 us\$ _____
 - b) 10.000 to 30.000 us\$ _____
 - c) 30.000 to 50.000 us\$ _____
 - d) more than 50.000 us\$ _____
2. Where do you usually go to buy supplies?
 - a. Hypermarket (big supermarket)
 - b. Medium-Small supermarket
 - c. Grocery shop
 - d. Street market
 - e. Other _____
3. What bothers you the most about grocery shopping? What else?

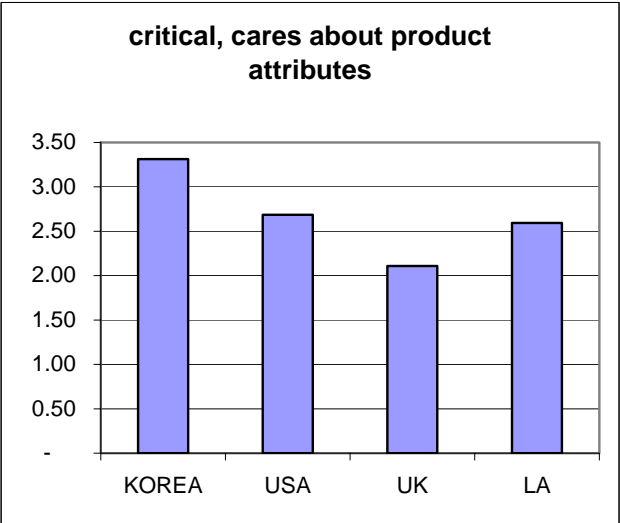
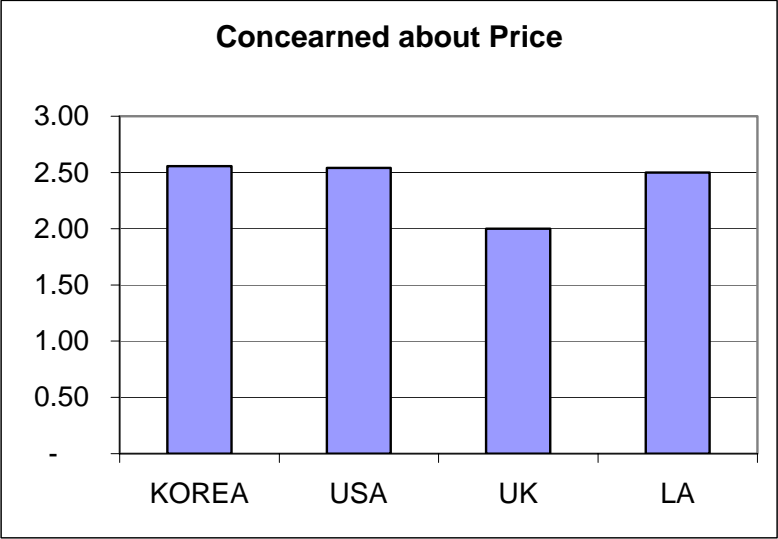
4. Do you think that the experience of shopping for groceries can be improved? How?

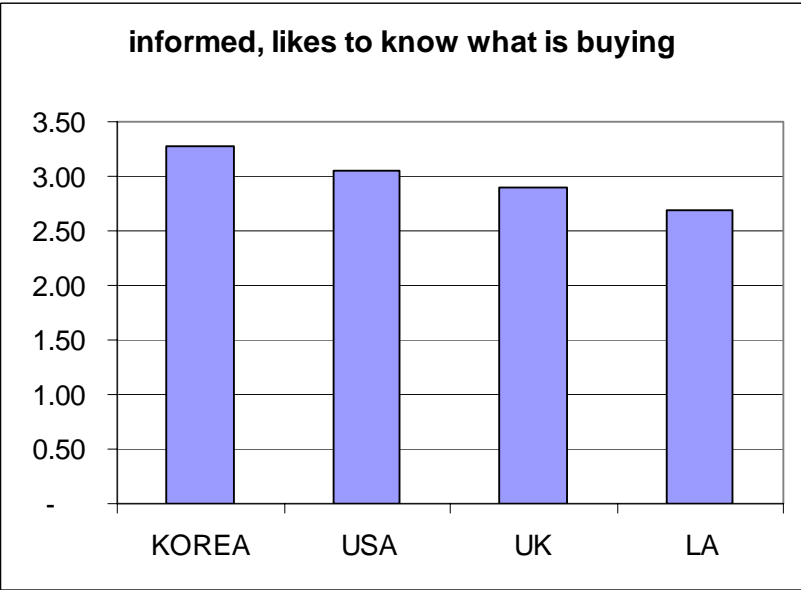
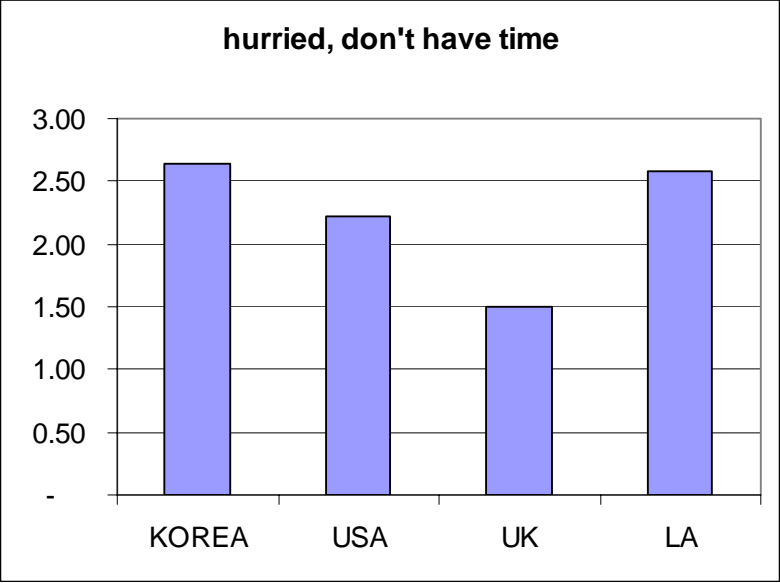
For the following statements regarding shopping for groceries please rank your agreement from 1 (Strongly Agree) to 5 (Strongly Disagree).

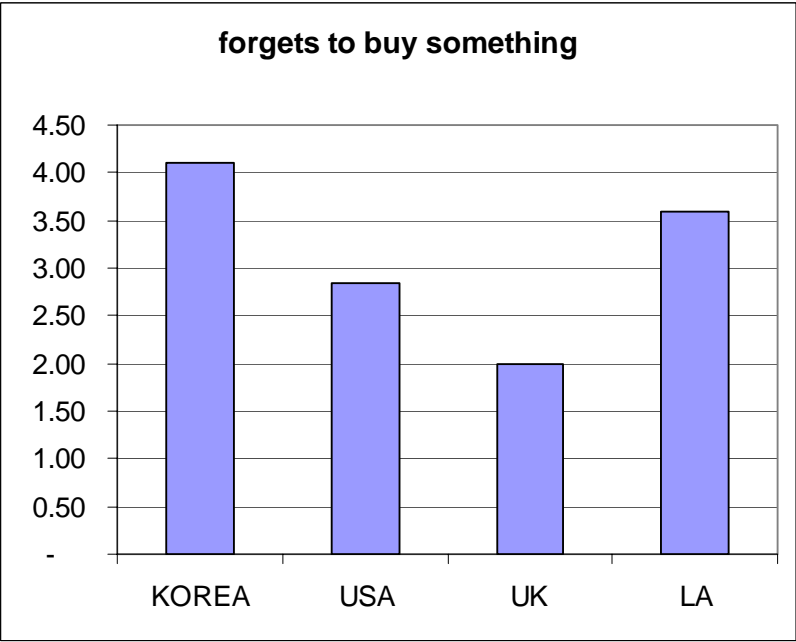
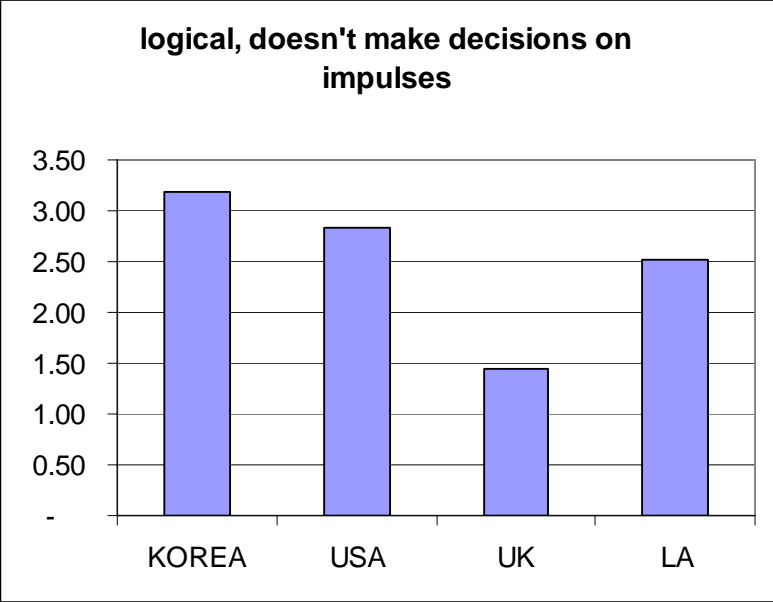
5. I know what I need, I go straight to it, grab everything I need, pay and leave as soon as possible. _____
6. I don't buy some products because I don't like the manufacturer _____
7. I take some time to compare prices, check for offers or promotions and compare some brands. _____
8. I take some time to check for ingredients, nutritional information and/or other information that may be on the product label. _____
9. I always buy the same brands _____
10. It's hard to locate some items. Either I spend too much time looking for it or ask for help _____
11. I don't buy genetically modified food _____
12. I need to check the ingredients on all products because I have a special dietary need _____
13. There are some brands that I never buy _____
14. It is very difficult to compare prices between brands _____
15. I buy products that have been tested on animals _____
16. It is very easy to decide between two different branded products _____
17. Waiting time for paying is very long _____
18. I pick a product because I like the packaging. _____
19. I always forget to buy something I needed _____
20. It's hard for me to identify the price of the products _____
21. I decide what to buy while I tour the supermarket _____
22. I like to shop in small groceries stores where I know some of the employees and I can talk to them. _____

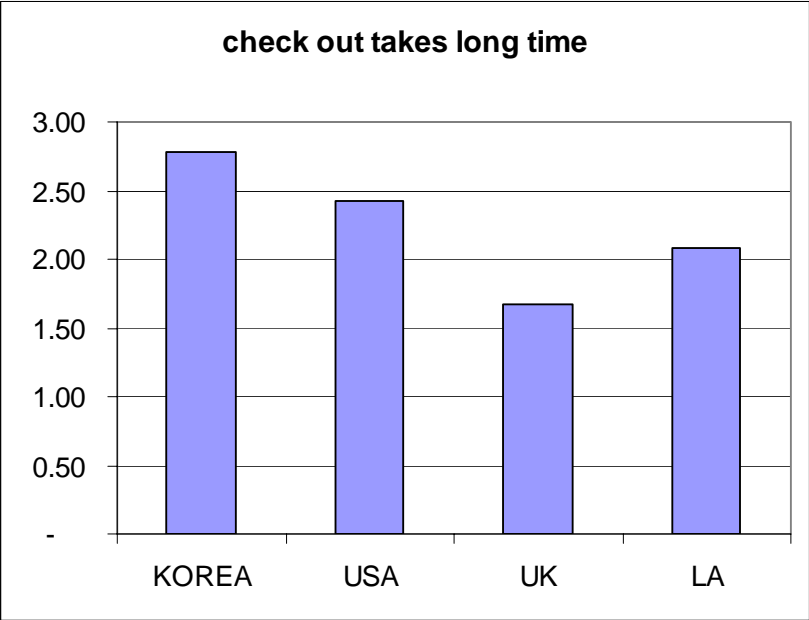
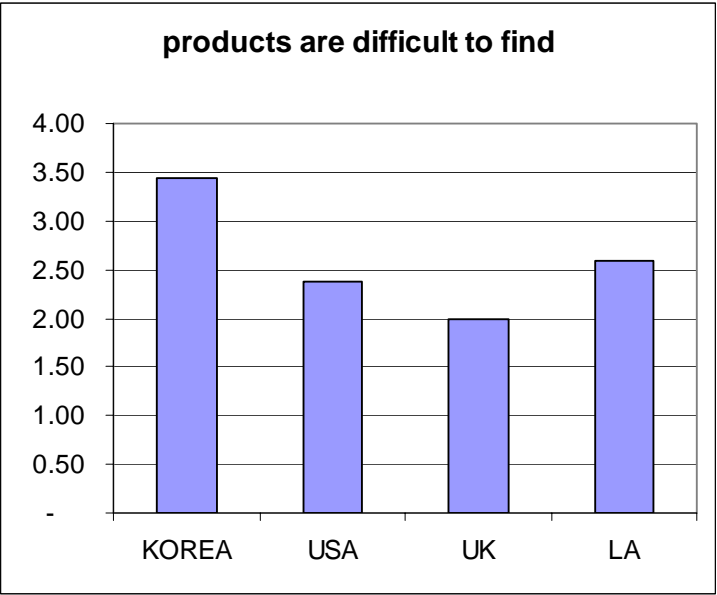
APPENDIX 5 – REGIONAL CONSUMER CONCERNS

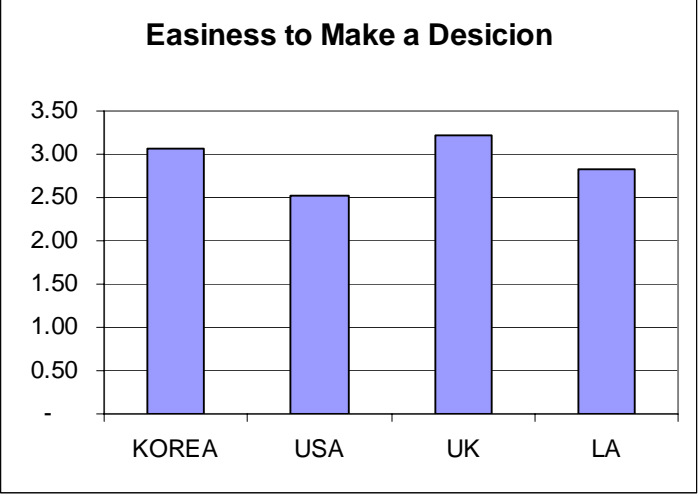




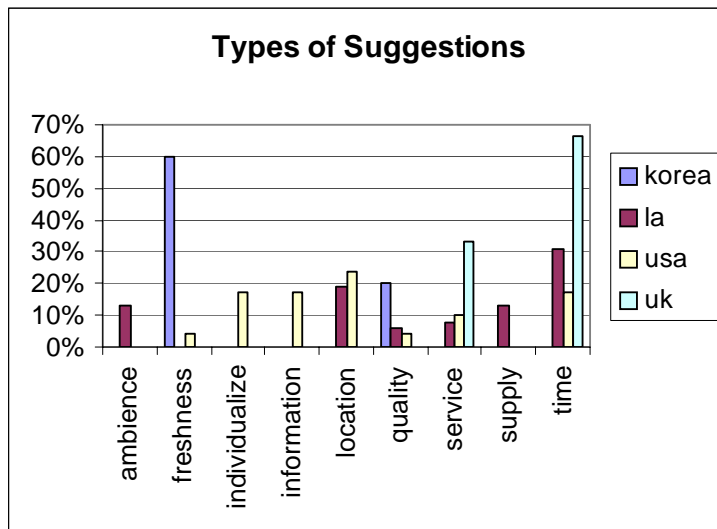
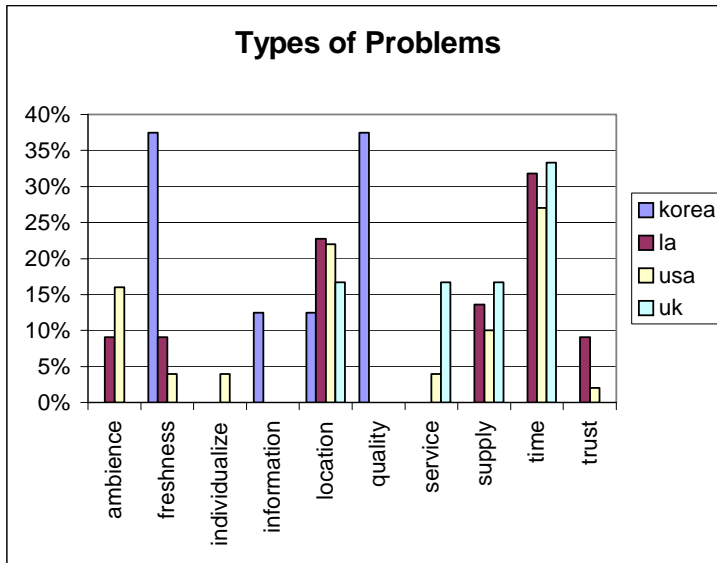




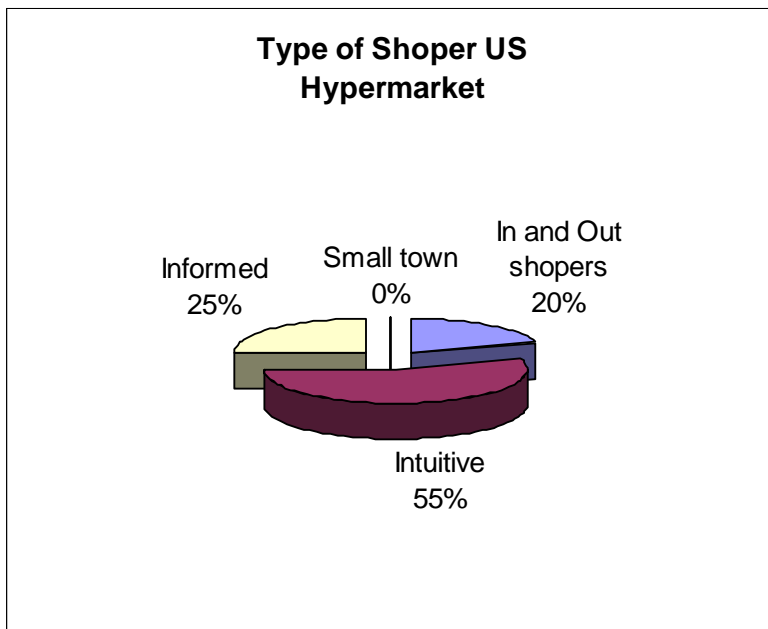
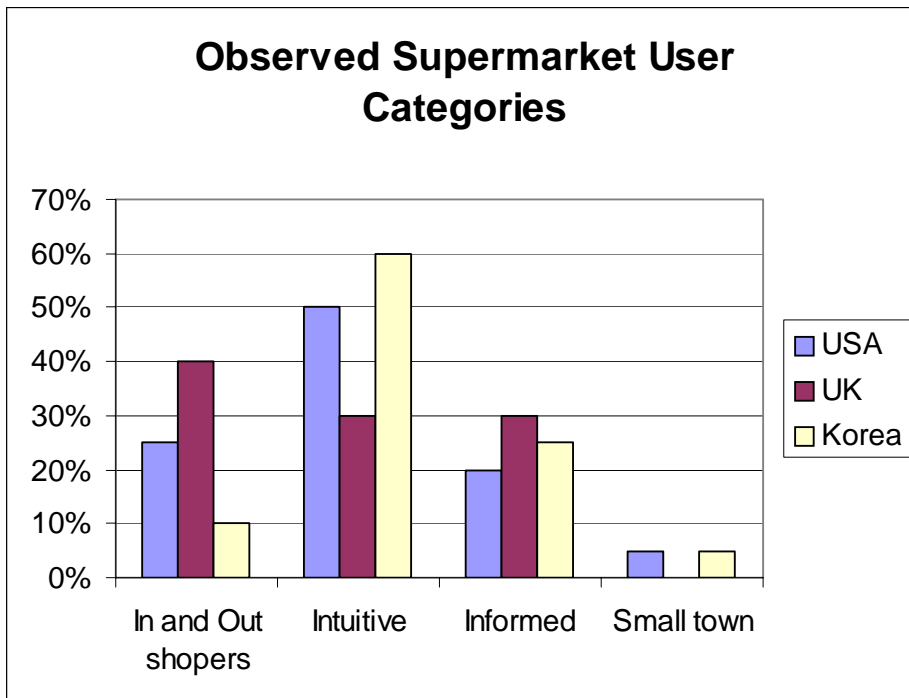




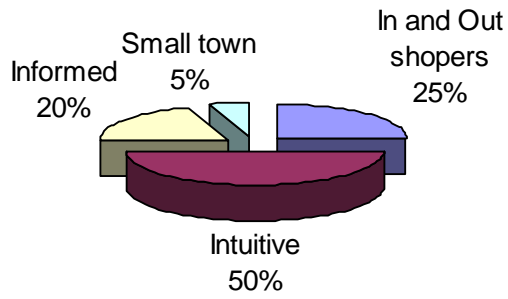
APPENDIX 6 – REGIONAL CONSUMER ATTITUDES



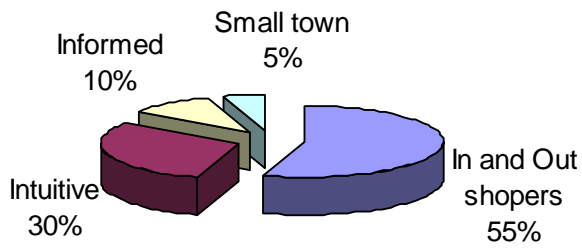
APPENDIX 7 – GENERAL REGIONAL OBSERVATIONS



**Type of Shoper US
Small-Medium Supermarkets**



**Type of Shoper US
Groceries stores**



APPENDIX 8 – AWARE™ SOFTWARE CODE

Header Files

- **Aware.h** : header for all source functions and constants
- **AwareRsc.h** : header defining resources for GUI -> generated by Constructor GUI builder program (a part of CodeWarrior for PalmOS).

```
/*
 * File: Aware.h
 *
 */
// #define SCANCODE 1 // Activates all of the CSM-150 scanner code

// Name the ingredients in the database
#define EGG 0
#define BUTTERMILK 1
#define CREAM 2
#define ARTIFICIAL_COLOR 3
#define YELLOW5 4
#define PRESERVATIVES 5
#define SALT 6
#define ASPIRIN 7
#define ONION 8
#define VINEGAR 9
#define NUM_INGREDIENTS 10 // (see initializeIngredients function)

// Ingredient preference states
#define NO_PREF_INGR 0
#define PREFER_INGR 1
#define BAN_INGR 2

// Name the products in the database
// Make sure the order is the same as in the list in InfoForm
#define RANCH_HIDDEN 0
#define RANCH_KRAFT 1
#define GODDESS_ANNIES 2
#define KETCHUP_HEINZ 3
#define KETCHUP_HUNTZ 4
#define KETCHUP_MEIJER 5
#define NUM_PRODUCTS 6 // Total number of products

// Product Types
#define SALAD_DRESSING 0
#define KETCHUP 1

// PalmOS standard libraries
#include <PalmOS.h> // Standard PalmOS libraries
#include "PalmCompatibility.h" // These 2 together replace Pilot.h

// Handspring-specific headers
#include "HsExt.h" // Handspring Common Includes

// Scanner-specific headers
#ifdef SCANCODE
#include "ScanMgrDef.h"
#include "ScanMgrStruct.h"
#include "ScanMgr.h"
#endif

/*
 * Internal Structures
 */
typedef struct productDB {
    Int8 ID;
    Int8 type;
    char name[50];
    char brand[25];
    char size[20];
    char price[20];
    char unitprice[20];
    char alerts[256];
    Boolean hasIngredient[NUM_INGREDIENTS];
    char barcode[20];
} productDB;
typedef productDB *productDBPtr;

/*
 * Global Variables
 */

extern char currentScan[];
extern productDB product[];
extern char ingredientName[NUM_INGREDIENTS][25];
extern Int8 ingredientPref[NUM_INGREDIENTS];
//extern Boolean ingredientAlertMode;
```

```

/*****
 *
 * Function Prototypes
 *
 *****/

/* Functions in Aware.c */
void alertCheck(Int8 productID, char *alerts);
void setupIngredientNames(void);
void setupProducts(void);
void* GetObjectPtr(UInt16 objectID);
Err AppStart(void);
Err AppStop(void);
Boolean AppHandleEvent(EventPtr eventP);
void AppEventLoop(void);
UInt32 StarterPalmMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags);
UInt32 PilotMain( UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags);

/* Functions in Main.c */
Boolean MainFormHandleEvent(EventPtr eventP);
#ifdef SCANCODE
Boolean OnDecoderData();
#endif

/* Functions in Settings.c */
Boolean SetIngredientsFormHandleEvent(EventPtr eventP);

/* Functions in Info.c */
Boolean InfoFormHandleEvent(EventPtr eventP);
void fillInfo(void);

/* Functions in ID.c */
Boolean IDFormHandleEvent(EventPtr eventP);

```

```

//      Header generated by Constructor for Palm OS` 1.6
//
//      Generated at 5:58:48 PM on Wednesday, December 05, 2001
//
//      Generated for file: C:\Personal\J\Code\AWARE 1.5\Rsc\Aware.rsrc
//
//      THIS IS AN AUTOMATICALLY GENERATED HEADER FILE FROM CONSTRUCTOR FOR PALM OS";
//      - DO NOT EDIT - CHANGES MADE TO THIS FILE WILL BE LOST
//
//      Palm App Name:           "Untitled"
//
//      Palm App Version:        "1.0"

//      Resource: tFRM 1000
#define MainForm                1000      //(Left Origin = 0, Top Origin = 0, Width = 160, Height
= 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID = 0, Default Button ID = 0)
#define MainCompareButton      1013      //(Left Origin = 125, Top Origin = 59, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainScanButton        1014      //(Left Origin = 14, Top Origin = 59, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainInfoButton        1017      //(Left Origin = 70, Top Origin = 58, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainPrefsButton       1018      //(Left Origin = 15, Top Origin = 109, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainListButton        1019      //(Left Origin = 72, Top Origin = 109, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainCartButton        1020      //(Left Origin = 126, Top Origin = 109, Width = 22,
Height = 22, Usable = 1, Anchor Left = 1, Frame = 0, Non-bold Frame = 1, Font = Standard)
#define MainCompareBitmap     1200      //(Left Origin = 125, Top Origin = 58, Bitmap Resource
ID = 1200, Usable = 1)
#define MainScanBitmap        1100      //(Left Origin = 14, Top Origin = 60, Bitmap Resource
ID = 1100, Usable = 1)
#define MainInfoBitmap        1300      //(Left Origin = 69, Top Origin = 58, Bitmap Resource
ID = 1300, Usable = 1)
#define MainPrefsBitmap       1400      //(Left Origin = 15, Top Origin = 109, Bitmap Resource
ID = 1400, Usable = 1)
#define MainListBitmap        1500      //(Left Origin = 72, Top Origin = 109, Bitmap Resource
ID = 1500, Usable = 1)
#define MainCartBitmap        1600      //(Left Origin = 126, Top Origin = 108, Bitmap Resource
ID = 1600, Usable = 1)
#define MainPrefsLabel        1003      //(Left Origin = 16, Top Origin = 132, Usable = 1, Font
= Standard)
#define MainUnnamed1004Label   1004      //(Left Origin = 64, Top Origin = 132, Usable = 1, Font
= Standard)
#define MainUnnamed1005Label   1005      //(Left Origin = 75, Top Origin = 143, Usable = 1, Font
= Standard)
#define MainUnnamed1006Label   1006      //(Left Origin = 129, Top Origin = 132, Usable = 1,
Font = Standard)
#define MainScanLabel         1007      //(Left Origin = 16, Top Origin = 82, Usable = 1, Font
= Standard)
#define MainUnnamed1008Label   1008      //(Left Origin = 73, Top Origin = 81, Usable = 1, Font
= Standard)
#define MainUnnamed1009Label   1009      //(Left Origin = 117, Top Origin = 81, Usable = 1, Font
= Standard)
#define MainUnnamed1015Label   1015      //(Left Origin = 15, Top Origin = 19, Usable = 1, Font
= Bold 12)
#define MainUnnamed1016Label   1016      //(Left Origin = 25, Top Origin = 34, Usable = 1, Font
= Bold 12)

//      Resource: tFRM 1300
#define SetIngredientsForm     1300      //(Left Origin = 0, Top Origin = 0, Width = 160, Height
= 160, Usable = 1, Modal = 0, Save Behind = 1, Help ID = 0, Menu Bar ID = 0, Default Button ID = 0)
#define SetIngredientsResetAllButton 1306      //(Left Origin = 121, Top Origin = 144, Width = 38,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define SetIngredientsHomeButton 1311      //(Left Origin = 129, Top Origin = 1, Width = 30,
Height = 10, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define SetIngredientsPreferenceLabel 1302      //(Left Origin = 131, Top Origin = 71, Usable = 1, Font
= Standard)
#define SetIngredientsAlertsLabel 1313      //(Left Origin = 128, Top Origin = 21, Usable = 1, Font
= Standard)
#define SetIngredientsListList 1309      //(Left Origin = 3, Top Origin = 34, Width = 115,
Usable = 1, Font = Standard, Visible Items = 8)
#define SetIngredientsSettingsChoicesList 1312      //(Left Origin = 1, Top Origin = 17, Width = 124,
Usable = 0, Font = Standard, Visible Items = 8)
#define SetIngredientsSpecialNeedsList 1317      //(Left Origin = 41, Top Origin = 103, Width = 118,
Usable = 0, Font = Standard, Visible Items = 5)
#define SetIngredientsPreferPushButton 1303      //(Left Origin = 124, Top Origin = 97, Width = 33,
Height = 12, Usable = 1, Group ID = 2, Font = Standard)
#define SetIngredientsNonePushButton 1304      //(Left Origin = 124, Top Origin = 84, Width = 33,
Height = 12, Usable = 1, Group ID = 2, Font = Standard)
#define SetIngredientsBanPushButton 1305      //(Left Origin = 124, Top Origin = 110, Width = 33,
Height = 12, Usable = 1, Group ID = 2, Font = Standard)
#define SetIngredientsViewAsListPushButton 1307      //(Left Origin = 3, Top Origin = 123, Width = 56,
Height = 12, Usable = 1, Group ID = 3, Font = Standard)
#define SetIngredientsViewByClassPushButton 1308      //(Left Origin = 60, Top Origin = 123, Width = 58,
Height = 12, Usable = 1, Group ID = 3, Font = Standard)
#define SetIngredientsAlertOnPushButton 1314      //(Left Origin = 124, Top Origin = 34, Width = 33,
Height = 12, Usable = 1, Group ID = 1, Font = Standard)
#define SetIngredientsAlertsOffPushButton 1315      //(Left Origin = 124, Top Origin = 47, Width = 33,
Height = 12, Usable = 1, Group ID = 1, Font = Standard)
#define SetIngredientsPresetSpecialNeedsPopTrigger 1301      //(Left Origin = 0, Top Origin = 140, Width = 119,
Height = 19, Usable = 1, Anchor Left = 1, Font = Standard, List ID = 1317)
#define SetIngredientsSettingsChoicePopTrigger 1310      //(Left Origin = 0, Top Origin = 15, Width = 124,
Height = 17, Usable = 1, Anchor Left = 1, Font = Standard, List ID = 1312)
#define SetIngredientsGroupID 2
#define SetIngredientsGroupID2 3
#define SetIngredientsGroupID3 1

//      Resource: tFRM 1200
#define InfoForm              1200      //(Left Origin = 0, Top Origin = 0, Width = 160, Height
= 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID = 0, Default Button ID = 0)

```

```

#define InfoBuyButton 1203 //((Left Origin = 3, Top Origin = 146, Width = 22,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define InfoSuggestAlternativeButton 1204 //((Left Origin = 30, Top Origin = 146, Width = 54,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define InfoCompareButton 1206 //((Left Origin = 89, Top Origin = 146, Width = 42,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define InfoInfoButton 1207 //((Left Origin = 136, Top Origin = 146, Width = 22,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define InfoHomeButton 1208 //((Left Origin = 129, Top Origin = 1, Width = 30,
Height = 10, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define InfoAlertsField 1201 //((Left Origin = 38, Top Origin = 86, Width = 122,
Height = 59, Usable = 1, Editable = 1, Underline = 0, Single Line = 0, Dynamic Size = 1, Left Justified = 1, Max
Characters = 256, Font = Standard, Auto Shift = 1, Has Scroll Bar = 1, Numeric = 0)
#define InfoSizeField 1213 //((Left Origin = 38, Top Origin = 44, Width = 121,
Height = 13, Usable = 1, Editable = 1, Underline = 0, Single Line = 1, Dynamic Size = 0, Left Justified = 1, Max
Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define InfoPriceField 1214 //((Left Origin = 38, Top Origin = 58, Width = 122,
Height = 14, Usable = 1, Editable = 1, Underline = 0, Single Line = 1, Dynamic Size = 0, Left Justified = 1, Max
Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define InfoUnitPriceField 1216 //((Left Origin = 37, Top Origin = 72, Width = 123,
Height = 14, Usable = 1, Editable = 1, Underline = 0, Single Line = 1, Dynamic Size = 0, Left Justified = 1, Max
Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define InfoBrandField 1218 //((Left Origin = 38, Top Origin = 30, Width = 121,
Height = 13, Usable = 1, Editable = 1, Underline = 0, Single Line = 1, Dynamic Size = 0, Left Justified = 1, Max
Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll Bar = 0, Numeric = 0)
#define InfoUnnamed1202Label 1202 //((Left Origin = 1, Top Origin = 16, Usable = 1, Font =
Bold)
#define InfoUnitPriceLabel 1205 //((Left Origin = 1, Top Origin = 72, Usable = 1, Font =
Bold)
#define InfoUnnamed1211Label 1211 //((Left Origin = 1, Top Origin = 44, Usable = 1, Font =
Bold)
#define InfoUnnamed1212Label 1212 //((Left Origin = 1, Top Origin = 58, Usable = 1, Font =
Bold)
#define InfoUnnamed1215Label 1215 //((Left Origin = 0, Top Origin = 86, Usable = 1, Font =
Bold)
#define InfoUnnamed1217Label 1217 //((Left Origin = 1, Top Origin = 30, Usable = 1, Font =
Bold)
#define InfoProductList 1210 //((Left Origin = 38, Top Origin = 17, Width = 121,
Usable = 0, Font = Standard, Visible Items = 5)
#define InfoNamePopTrigger 1209 //((Left Origin = 33, Top Origin = 16, Width = 127,
Height = 15, Usable = 1, Anchor Left = 1, Font = Standard, List ID = 1210)

// Resource: tFRM 1400
#define IDForm 1400 //((Left Origin = 0, Top Origin = 0, Width = 160, Height
= 160, Usable = 1, Modal = 0, Save Behind = 0, Help ID = 0, Menu Bar ID = 0, Default Button ID = 0)
#define IDFindButton 1403 //((Left Origin = 33, Top Origin = 108, Width = 36,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define IDCancelButton 1404 //((Left Origin = 96, Top Origin = 108, Width = 36,
Height = 12, Usable = 1, Anchor Left = 1, Frame = 1, Non-bold Frame = 1, Font = Standard)
#define IDNumberField 1402 //((Left Origin = 30, Top Origin = 71, Width = 100,
Height = 12, Usable = 1, Editable = 1, Underline = 1, Single Line = 0, Dynamic Size = 0, Left Justified = 1, Max
Characters = 80, Font = Standard, Auto Shift = 0, Has Scroll Bar = 0, Numeric = 1)
#define IDEnterIDLabel 1401 //((Left Origin = 31, Top Origin = 32, Usable = 1, Font
= Standard)

// Resource: Talt 1000
#define NotImplementedAlert 1000
#define NotImplementedOK 0

// Resource: PICT 1000
#define BitmapID1000Bitmap 1000

// Resource: PICT 1100
#define BitmapID1100Bitmap 1100

// Resource: PICT 1200
#define BitmapID1200Bitmap 1200

// Resource: PICT 1300
#define BitmapID1300Bitmap 1300

// Resource: PICT 1400
#define BitmapID1400Bitmap 1400

// Resource: PICT 1500
#define BitmapID1500Bitmap 1500

// Resource: PICT 1600
#define BitmapID1600Bitmap 1600

// Resource: tbfm 1000
#define BitmapResourceID1000BitmapFamily 1000

// Resource: tbfm 1100
#define BitmapResourceID1100BitmapFamily 1100

// Resource: tbfm 1200
#define BitmapResourceID1200BitmapFamily 1200

// Resource: tbfm 1300
#define BitmapResourceID1300BitmapFamily 1300

// Resource: tbfm 1400
#define BitmapResourceID1400BitmapFamily 1400

// Resource: tbfm 1500
#define BitmapResourceID1500BitmapFamily 1500

// Resource: tbfm 1600

```

```
#define BitmapResourceID1600BitmapFamily    1600

//      Resource: taif 1000
#define LargeAppIconAppIconFamily          1000
```

Source Files:

- **Aware.c** : main source with initialization functions
- **Main.c** : event handler for main screen
- **Settings.c** : event handler for settings/pref screen
- **ID.c** : event handler for scan ID screen
- **Info.c** : event handler for product info screen

```

/*****
 * File: Aware.c
 *
 * Main file with initialization functions
 *
 *****/

#include "Aware.h"           // Includes all function headers for Aware
#include "AwareRsc.h"       // Includes GUI reference numbers

char currentScan[100];

/*****
 *
 * Internal Constants
 *
 *****/

// This is not the way databases are supposed to be handled, I think.
// I think there should be one database per Creator, but I can't figure
// out how to get it to work, so I created two separate databases.
// One is for storing ingrDB objects, and one for productDB objects.
#define ADBIngrCreator 'AwareI'
#define ADBProdCreator 'AwareP'

#define ADBType 'DATA'
#define ADBNameIngr "IngrAware"
#define ADBNameProd "ProdAware"

/*****
 *
 * Global Variables
 *
 *****/
productDB product[NUM_PRODUCTS];
char ingredientName[NUM_INGREDIENTS][25];
Int8 ingredientPref[NUM_INGREDIENTS];
//Boolean ingredientAlertMode;

/*****
 *
 * FUNCTION:   GetObjectPtr
 *
 * DESCRIPTION: This routine returns a pointer to an object in the current
 *              form.
 *
 * PARAMETERS: formId - id of the form to display
 *
 * RETURNED:   void* - pointer to the object on the form.
 *
 *****/
void* GetObjectPtr(UInt16 objectID) {
    FormPtr frmP;

    frmP = FrmGetActiveForm();
    return FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID));
}

/*****
 *
 * FUNCTION:   AppStart
 *
 * DESCRIPTION: Create or open the program's database.
 *
 * PARAMETERS: none.
 *
 * RETURNED:   error - 0 if the database opened fine or the error
 *              code produced if there was a problem.
 *
 *****/
Err AppStart(void) {
    Err error = 0;
    Int8 i;

```

```

    for(i=0; i<NUM_INGREDIENTS; i++) {
        ingredientPref[i] = 0;
    }
    setupIngredientNames(); // Stored as a global array
    setupProducts(); // Stored as global structures
    //ingredientAlertMode = true;

    // SCANNER
    #ifdef SCANCODE
    if(ScanIsPalmSymbolUnit()) {
        // Open scan manager library
        error = ScanOpenDecoder();
        // Set decoder parameters we care about
        ScanCmdScanEnable(); // enable scanning
        // Allow software triggered scans
        // ScanSetTriggeringModes( HOST );
        // Enable barcode types
        ScanSetBarcodeEnabled( barUPCA, true );
        ScanSetBarcodeEnabled( barUPCE, true );
        // Send these parameters to the decoder
        ScanCmdSendParams( No_Beep );
    }
    #endif

    return error;
}

/*****
 *
 * FUNCTION: AppStop
 *
 * DESCRIPTION: Save the current state of the application.
 *
 * PARAMETERS: nothing
 *
 * RETURNED: error - 0 if there was no problem closing the database,
 * and an error code if there was a problem.
 *****/
Err AppStop(void) {
    // Close all the open forms.
    FrmCloseAllForms ();

    // Shut down scanner
    #ifdef SCANCODE
    if (ScanIsPalmSymbolUnit()) {
        ScanCmdScanDisable();
        ScanCloseDecoder();
    }
    #endif

    return 0;
}

/*****
 *
 * FUNCTION: AppHandleEvent
 *
 * DESCRIPTION: This routine loads form resources and set the event
 * handler for the form loaded.
 *
 * PARAMETERS: event - a pointer to an EventType structure
 *
 * RETURNED: true if the event has handle and should not be passed
 * to a higher level handler.
 *****/
Boolean AppHandleEvent(EventPtr eventP) {
    UInt16 formId;
    FormPtr frmP;

    if (eventP->eType == frmLoadEvent) {
        /* Load the form resource. */
        formId = eventP->data.frmLoad.formID;
        frmP = FrmInitForm(formId);
        FrmSetActiveForm(frmP);

        /* Set the event handler for the form. The handler of the currently
        * active form is called by FrmHandleEvent each time it receives an
        * event. */
        switch (formId) {
            case MainForm:
                FrmSetEventHandler(frmP, MainFormHandleEvent);
                break;
            case SetIngredientsForm:
                FrmSetEventHandler(frmP, SetIngredientsFormHandleEvent);
                break;
            case IDForm:
                FrmSetEventHandler(frmP, IDFormHandleEvent);
                break;
            case InfoForm:
                FrmSetEventHandler(frmP, InfoFormHandleEvent);
                break;
            default:
                break;
        }
        return true;
    }
    return false;
}

```

```

/*****
 *
 * FUNCTION:   AppEventLoop
 *
 * DESCRIPTION: This routine is the event loop for the application.
 *
 * PARAMETERS: nothing
 *
 * RETURNED:  nothing
 *
 *****/
void AppEventLoop(void) {
    UInt16 error;
    EventType event;

    do {
        EvtGetEvent(&event, evtWaitForever);

        // Try to handle the event by these systems in order.
        // If an event is not handled by one system, it will
        // return "false" and the next handler in the chain
        // will be called. Sys->Menu->App->FrmDispatch
        if (! SysHandleEvent(&event))
            if (! MenuHandleEvent(0, &event, &error))
                if (! AppHandleEvent(&event))
                    FrmDispatchEvent(&event);

    } while (event.eType != appStopEvent);
}

/*****
 *
 * FUNCTION:   StarterPalmMain
 *
 * DESCRIPTION: This is the main entry point for the application.
 *
 * PARAMETERS: cmd - word value specifying the launch code.
 *             cmdPB - pointer to a structure that is associated with the launch code.
 *             launchFlags - word value providing extra information about the launch.
 *
 * RETURNED:  Result of launch
 *
 *****/
UInt32 StarterPalmMain(UInt16 cmd, MemPtr /*cmdPBP*/, UInt16 launchFlags) {
    Err error;

    switch (cmd) {
        case sysAppLaunchCmdNormalLaunch:
            error = AppStart();
            if (error)
                return error;
            FrmGotoForm(MainForm);
            AppEventLoop();
            error = AppStop();
            if (error)
                return error;
            break;

        default:
            break;
    }
    return errNone;
}

/*****
 *
 * FUNCTION:   PilotMain
 *
 * DESCRIPTION: This is the main entry point for the application.
 *
 * PARAMETERS: cmd - word value specifying the launch code.
 *             cmdPB - pointer to a structure that is associated with the launch code.
 *             launchFlags - word value providing extra information about the launch.
 *
 * RETURNED:  Result of launch
 *
 * REVISION HISTORY: Taken from Starter.c stationery project, copyright
 *                   (c) 1999 Palm Computing Inc. or its subsidiaries. All
 *                   rights reserved.
 *
 *****/
UInt32 PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags) {
    return StarterPalmMain(cmd, cmdPBP, launchFlags);
}

/*****
 *
 * FUNCTION:   setupProducts
 *
 * DESCRIPTION: Sets up the product names and information (hard coded
 *             for simplicity - should really be a database)
 *
 *
 *
 *****/
void setupProducts(void) {
    int i;

    // RANCH DRESSING: HIDDEN VALLEY

```



```

product[RANCH_HIDDEN].type = SALAD_DRESSING;
StrCopy(product[RANCH_HIDDEN].barcode, "07155019");
StrCopy(product[RANCH_HIDDEN].name, "Ranch Dressing (Hidden Valley)");
StrCopy(product[RANCH_HIDDEN].brand, "Hidden Valley, USA");
StrCopy(product[RANCH_HIDDEN].size, "8 oz.");
StrCopy(product[RANCH_HIDDEN].price, "$2.09");
StrCopy(product[RANCH_HIDDEN].unitprice, "$0.2613 per oz.");
StrCopy(product[RANCH_HIDDEN].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[RANCH_HIDDEN].hasIngredient[0] = false;
// Then, designate ingredients that are in the product
product[RANCH_HIDDEN].hasIngredient[EGG] = true;
product[RANCH_HIDDEN].hasIngredient[ARTIFICIAL_COLOR] = true;
product[RANCH_HIDDEN].hasIngredient[BUTTERMILK] = true;
product[RANCH_HIDDEN].hasIngredient[CREAM] = true;
product[RANCH_HIDDEN].hasIngredient[SALT] = true;
product[RANCH_HIDDEN].hasIngredient[PRESERVATIVES] = true;
product[RANCH_HIDDEN].hasIngredient[VINEGAR] = true;

// RANCH DRESSING: KRAFT
product[RANCH_KRAFT].type = SALAD_DRESSING;
StrCopy(product[RANCH_KRAFT].barcode, "021000643592");
StrCopy(product[RANCH_KRAFT].name, "Ranch Dressing (Kraft)");
StrCopy(product[RANCH_KRAFT].brand, "Kraft, USA");
StrCopy(product[RANCH_KRAFT].size, "8 oz.");
StrCopy(product[RANCH_KRAFT].price, "$1.79");
StrCopy(product[RANCH_KRAFT].unitprice, "$0.2238 per oz.");
StrCopy(product[RANCH_KRAFT].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[RANCH_KRAFT].hasIngredient[0] = false;
// Then, designate ingredients that are in the product
product[RANCH_KRAFT].hasIngredient[EGG] = true;
product[RANCH_KRAFT].hasIngredient[ARTIFICIAL_COLOR] = true;
product[RANCH_KRAFT].hasIngredient[BUTTERMILK] = true;
product[RANCH_KRAFT].hasIngredient[CREAM] = true;
product[RANCH_KRAFT].hasIngredient[SALT] = true;
product[RANCH_KRAFT].hasIngredient[PRESERVATIVES] = true;
product[RANCH_KRAFT].hasIngredient[VINEGAR] = true;

// GODDESS DRESSING: ANNIE'S NATURALS
product[GODDESS_ANNIES].type = SALAD_DRESSING;
StrCopy(product[GODDESS_ANNIES].barcode, "092325333369");
StrCopy(product[GODDESS_ANNIES].name, "Goddess Dressing (Annie's Naturals)");
StrCopy(product[GODDESS_ANNIES].brand, "Annie's Naturals, USA");
StrCopy(product[GODDESS_ANNIES].size, "8 oz.");
StrCopy(product[GODDESS_ANNIES].price, "$2.89");
StrCopy(product[GODDESS_ANNIES].unitprice, "$0.3613 per oz.");
StrCopy(product[GODDESS_ANNIES].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[GODDESS_ANNIES].hasIngredient[0] = false;
// Then, designate ingredients that are in the product
product[GODDESS_ANNIES].hasIngredient[VINEGAR] = true;

// KETCHUP: HEINZ
product[KETCHUP_HEINZ].type = KETCHUP;
StrCopy(product[KETCHUP_HEINZ].barcode, "01312403");
StrCopy(product[KETCHUP_HEINZ].name, "Tomato Ketchup (Heinz)");
StrCopy(product[KETCHUP_HEINZ].brand, "Heinz, USA");
StrCopy(product[KETCHUP_HEINZ].size, "14 oz.");
StrCopy(product[KETCHUP_HEINZ].price, "$1.29");
StrCopy(product[KETCHUP_HEINZ].unitprice, "$0.0921 per oz.");
StrCopy(product[KETCHUP_HEINZ].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[KETCHUP_HEINZ].hasIngredient[0] = false;
// Then, designate ingredients that are in the product
product[KETCHUP_HEINZ].hasIngredient[VINEGAR] = true;
product[KETCHUP_HEINZ].hasIngredient[SALT] = true;

// KETCHUP: HUNT'S
product[KETCHUP_HUNTZ].type = KETCHUP;
StrCopy(product[KETCHUP_HUNTZ].barcode, "027000382493");
StrCopy(product[KETCHUP_HUNTZ].name, "Tomato Ketchup (Hunt's)");
StrCopy(product[KETCHUP_HUNTZ].brand, "Hunt's, USA");
StrCopy(product[KETCHUP_HUNTZ].size, "24 oz.");
StrCopy(product[KETCHUP_HUNTZ].price, "$1.39");
StrCopy(product[KETCHUP_HUNTZ].unitprice, "$0.0579 per oz.");
StrCopy(product[KETCHUP_HUNTZ].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[KETCHUP_HUNTZ].hasIngredient[0] = false;
// Then, designate ingredients that are in the product
product[KETCHUP_HUNTZ].hasIngredient[VINEGAR] = true;
product[KETCHUP_HUNTZ].hasIngredient[SALT] = true;

// KETCHUP: MEIJER
product[KETCHUP_MEIJER].type = KETCHUP;
StrCopy(product[KETCHUP_MEIJER].barcode, "041250040088");
StrCopy(product[KETCHUP_MEIJER].name, "Tomato Ketchup (Meijer)");
StrCopy(product[KETCHUP_MEIJER].brand, "Meijer, USA");
StrCopy(product[KETCHUP_MEIJER].size, "14 oz.");
StrCopy(product[KETCHUP_MEIJER].price, "$0.99");
StrCopy(product[KETCHUP_MEIJER].unitprice, "$0.07071 per oz.");
StrCopy(product[KETCHUP_MEIJER].alerts, "<no alerts>");
// First, set all ingredients to default of false.
for (i=0; i<NUM_INGREDIENTS; i++) product[KETCHUP_MEIJER].hasIngredient[0] = false;
// Then, designate ingredients that are in the product

```

```

        product[KETCHUP_MEIJER].hasIngredient[VINEGAR] = true;
        product[KETCHUP_MEIJER].hasIngredient[SALT] = true;
    }

    /*****
    *
    * FUNCTION:      setupIngredientNames
    *
    * DESCRIPTION:  Sets up the ingredient names (hard coded
    *                for simplicity - should really be a database)
    *                NOTE: ingredient preference info is stored in database
    *                so that the information is retained if the program is
    *                closed.
    *
    *
    *
    *****/
    void setupIngredientNames(void) {
        StrCopy(ingredientName[EGG], "Egg");
        StrCopy(ingredientName[SALT], "Salt");
        StrCopy(ingredientName[ASPIRIN], "Aspirin");
        StrCopy(ingredientName[BUTTERMILK], "Buttermilk");
        StrCopy(ingredientName[ONION], "Onion");
        StrCopy(ingredientName[PRESERVATIVES], "Preservatives");
        StrCopy(ingredientName[ARTIFICIAL_COLOR], "Artificial Color");
        StrCopy(ingredientName[CREAM], "Cream");
        StrCopy(ingredientName[VINEGAR], "Vinegar");
        StrCopy(ingredientName[YELLOW5], "Yellow #5");
    }

```

```

/*****
 * File: Main.c
 *
 * Event handler for the main Aware form
 *
 *****/

#include <PalmOS.h>
#include "AwareRsc.h"
#include "Aware.h"

/*****
 *
 * FUNCTION:    MainFormHandleEvent
 *
 * DESCRIPTION: This routine is the event handler for the
 *              "MainForm" of this application.
 *
 * PARAMETERS: eventP - a pointer to an EventType structure
 *
 * RETURNED:   true if the event has handle and should not be passed
 *              to a higher level handler.
 *
 *****/
Boolean MainFormHandleEvent(EventPtr eventP) {
    Boolean handled = false;
    FormPtr frmP;
    //UInt16 numRecs = DmNumRecords(DB);

    switch (eventP->eType) {
        case frmOpenEvent: // Form is opened
            frmP = FrmGetActiveForm();
            FrmDrawForm (frmP);
            handled = true;
            break;

        case ctlSelectEvent: // User presses something
            switch (eventP->data.ctlSelect.controlID) {
                case MainScanButton:
                    // Try to scan product
                    #ifndef SCANCODE
                    if(ScanIsPalmSymbolUnit()) {
                        ScanCmdStartDecode();
                        handled = true;
                        break;
                    }
                    #endif
                    // If not successful, go to manual ID form
                    FrmGotoForm(IDForm);
                    handled = true;
                    break;
                case MainCompareButton:
                    FrmAlert(NotImplementedAlert);
                    handled = true;
                    break;
                case MainInfoButton:
                    FrmGotoForm(InfoForm);
                    handled = true;
                    break;
                case MainPrefsButton:
                    FrmGotoForm(SetIngredientsForm);
                    handled = true;
                    break;
                case MainListButton:
                    FrmAlert(NotImplementedAlert);
                    handled = true;
                    break;
                case MainCartButton:
                    FrmAlert(NotImplementedAlert);
                    handled = true;
                    break;
            }
            break;
        #ifndef SCANCODE
        case scanDecodeEvent:
            // A decode has been performed.
            // User decoder API to retrieve decoder data
            OnDecoderData();
            handled = true;
            break;
        #endif
    }
    return handled;
}

/*****
 *
 * FUNCTION:    OnDecoderData
 *
 *****/
#ifdef SCANCODE
Boolean OnDecoderData() {
    #ifndef SCANCODE
    static Char BarTypeStr[80];
    MESSAGE decodeDataMsg;
    //FieldType *fld = GetObjectPtr(MainDataField);

    int refNum = 0;

    int status = ScanGetDecodedData( &decodeDataMsg );
    #endif
}
#endif

```

```
// if we successfully got the decode data from the API...
if (status == STATUS_OK) {
    // Then write the data in the MainDataField
    if( StrNCompare((char*)decodeDataMsg.data,"NR",2)==0) {
        // No read
        FldDelete( fld, 0, FldGetMaxChars(fld));
        FldInsert(fld,"No Read", StrLen("No Read"));
    }
    else {
        // Successful read
        //SetFieldText( MainDataField,
        //              (char *)&decodeDataMsg.data[0], 79, true);
    }
}
#endif
return(0);
}
#endif
```

```

/*****
 * File: Settings.c
 *
 *****/
#define DO_NOT_ALLOW_ACCESS_TO_INTERNALS_OF_STRUCTS

#include <PalmOS.h>
#include "AwareRsc.h"
#include "Aware.h"

/*****
 *
 * FUNCTION: SetIngredientsFormHandleEvent
 *
 * DESCRIPTION: This routine is the event handler for the "IngredientsForm" of
 *              this application.
 *
 * PARAMETERS: eventP - a pointer to an EventType structure
 *
 * RETURNED:  handled - false if the event should still be handled by
 *              the system, true if the system should ignore it.
 *****/
Boolean SetIngredientsFormHandleEvent(EventPtr eventP) {
    Boolean handled = false;
    FormPtr frmP = FrmGetActiveForm();
    ListType *list = GetObjectPtr(SetIngredientsListList);
    UInt ingrPref;

    switch (eventP->eType) {
        case frmOpenEvent:
            CtlSetValue(GetObjectPtr(SetIngredientsViewAsListPushButton),1);
            CtlSetValue(GetObjectPtr(SetIngredientsAlertOnPushButton),1);
            // Update ingredient preference
            ingrPref = ingredientPref[LstGetSelection(list)];
            CtlSetValue(GetObjectPtr(SetIngredientsNonePushButton),(1-ingrPref)*(2-ingrPref));
            CtlSetValue(GetObjectPtr(SetIngredientsPreferPushButton),(0-ingrPref)*(2-ingrPref));
            CtlSetValue(GetObjectPtr(SetIngredientsBanPushButton),(0-ingrPref)*(1-ingrPref));
            // Update alert mode
            //CtlSetValue(GetObjectPtr(SetIngredientsAlertOnPushButton),(ingredientAlertMode));
            //CtlSetValue(GetObjectPtr(SetIngredientsAlertsOffPushButton),(1-
ingredientAlertMode));

            // Draw form
            frmP = FrmGetFormPtr(SetIngredientsForm);
            FrmDrawForm (frmP);
            handled = true;
            break;

        case ctlSelectEvent:
            switch (eventP->data.ctlSelect.controlID) {
                case SetIngredientsHomeButton:
                    FrmGotoForm(MainForm);
                    break;

                case SetIngredientsViewByClassPushButton:
                    FrmAlert(NotImplementedAlert);
                    // Update ingredient view PushButtons
                    CtlSetValue(GetObjectPtr(SetIngredientsViewByClassPushButton),0);
                    CtlSetValue(GetObjectPtr(SetIngredientsViewAsListPushButton),1);
                    break;

                case SetIngredientsNonePushButton:
                    ingredientPref[LstGetSelection(list)] = 0;
                    break;

                case SetIngredientsPreferPushButton:
                    ingredientPref[LstGetSelection(list)] = 1;
                    break;

                case SetIngredientsBanPushButton:
                    ingredientPref[LstGetSelection(list)] = 2;
                    break;

                case SetIngredientsResetAllButton:
                    for(ingrPref=0; ingrPref<NUM_INGREDIENTS; ingrPref++)
                        ingredientPref[ingrPref] = 0;
                    ingrPref = ingredientPref[LstGetSelection(list)];
                    CtlSetValue(GetObjectPtr(SetIngredientsNonePushButton),(1-
ingrPref)*(2-ingrPref));
                    CtlSetValue(GetObjectPtr(SetIngredientsPreferPushButton),(0-
ingrPref)*(2-ingrPref));
                    CtlSetValue(GetObjectPtr(SetIngredientsBanPushButton),(0-
ingrPref)*(1-ingrPref));
                    break;

                /*case SetIngredientsAlertOnPushButton:
                    ingredientAlertMode = true;
                    break;

                case SetIngredientsAlertsOffPushButton:
                    ingredientAlertMode = false;
                    break;*/
            }
            break;

        case popSelectEvent:
            switch (eventP->data.popSelect.listID) {
                case (SetIngredientsSpecialNeedsList):
                    switch (eventP->data.popSelect.selection) {
                        case 1: // Allergy to poultry
                            ingredientPref[EGG] = BAN_INGR;
                            break;

                        case 5: // Lactose intolerant
                            ingredientPref[CREAM] = BAN_INGR;
                            ingredientPref[BUTTERMILK] = BAN_INGR;
                            break;

                        case 6: // Vegan
                            ingredientPref[EGG] =
                            ingredientPref[BUTTERMILK] =
                            ingredientPref[CREAM] = BAN_INGR;
                    }
            }
    }
}

```

```

        break;
        case 7: // Vegetarian
            ingredientPref[EGG] = BAN_INGR;
            break;
    }
    break;
    case (SetIngredientsSettingsChoicesList):
        switch (eventP->data.popSelect.selection) {
            case 0: // Ingredients
                break;
            default:
                FrmAlert(NotImplementedAlert);
                break;
        }
        break;
    }
    // break;
    case lstSelectEvent:
        // Update Preference PushButtons to current ingredient pref setting
        ingrPref = ingredientPref[LstGetSelection(list)];
        CtlSetValue(GetObjectPtr(SetIngredientsNonePushButton), (1-ingrPref)*(2-ingrPref));
        CtlSetValue(GetObjectPtr(SetIngredientsPreferPushButton), (0-ingrPref)*(2-ingrPref));
        CtlSetValue(GetObjectPtr(SetIngredientsBanPushButton), (0-ingrPref)*(1-ingrPref));
        break;
    }
    return handled;
}

```

```

/*****
 * File: ID.c
 *
 * When scanner cannot find an ID, this form allows the
 * user to enter the barcode ID number manually.
 *****/

#include <PalmOS.h>
#include "AwareRsc.h"
#include "Aware.h"

/*****
 *
 * FUNCTION: IDFormHandleEvent
 *
 * DESCRIPTION: This routine is the event handler for the "IDForm" of
 * this application.
 *
 * PARAMETERS: eventP - a pointer to an EventType structure
 *
 * RETURNED: handled - false if the event should still be handled by
 * the system, true if the system should ignore it.
 *****/
Boolean IDFormHandleEvent(EventPtr eventP) {
    Boolean handled = false;
    FormPtr frmP;

    switch (eventP->eType) {
        case frmOpenEvent:
            frmP = FrmGetActiveForm();
            FrmDrawForm (frmP);
            handled = true;
            break;

        case ctlSelectEvent:
            switch (eventP->data.ctlSelect.controlID) {
                case IDFindButton:
                    // Take number from blank and look it up
                    // in the info form
                    StrCopy(currentScan, FldGetTextPtr(GetObjectPtr(IDNumberField)));
                    FrmGotoForm(InfoForm);
                    break;
                case IDCancelButton:
                    FrmGotoForm(MainForm);
                    break;
            }
            break;
    }
    return handled;
}

```

```

/*****
 * File: Info.c
 *
 *****/
#define DO_NOT_ALLOW_ACCESS_TO_INTERNALS_OF_STRUCTS

#include <PalmOS.h>
#include "AwareRsc.h"
#include "Aware.h"

/*****
 *
 * FUNCTION:    InfoFormHandleEvent
 *
 * DESCRIPTION: This routine is the event handler for the "IngredientsForm" of
 *              this application.
 *
 * PARAMETERS: eventP - a pointer to an EventType structure
 *
 * RETURNED:   handled - false if the event should still be handled by
 *              the system, true if the system should ignore it.
 *
 *****/
Boolean InfoFormHandleEvent(EventPtr eventP) {
    Int8 i;
    Boolean handled = false;
    Boolean preselect = false;
    FormPtr frmP = FrmGetActiveForm();
    ListType *list = GetObjectPtr(InfoProductList);

    switch (eventP->eType) {
        case frmOpenEvent:
            preselect = false;
            for (i=0; i<NUM_PRODUCTS; i++) {
                if (StrCompare(product[i].barcode, currentScan) == 0) {
                    LstSetSelection(list, i);
                    CtlSetLabel(GetObjectPtr(InfoNamePopTrigger), product[i].name);
                    preselect = true;
                    fillInfo();
                }
            }
            if (preselect == false)
                CtlSetLabel(GetObjectPtr(InfoNamePopTrigger), "<select>");
            FrmDrawForm (frmP);
            handled = true;
            break;
        case ctlSelectEvent:
            switch(eventP->data.ctlSelect.controlID) {
                case InfoHomeButton:
                    FrmGotoForm(MainForm);
                    break;
                case InfoBuyButton:
                    FrmAlert(NotImplementedAlert);
                    break;
                case InfoSuggestAlternativeButton:
                    if (LstGetSelection(list) < 2) {
                        if (ingredientPref[EGG] == BAN_INGR ||
                            ingredientPref[BUTTERMILK] == BAN_INGR ||
                            ingredientPref[CREAM] == BAN_INGR) {
                            LstSetSelection(list, GODDESS_ANNIES);
                            CtlSetLabel(GetObjectPtr(InfoNamePopTrigger),
                                product[GODDESS_ANNIES].name);
                            fillInfo();
                        }
                    }
                    break;
                case InfoCompareButton:
                    FrmAlert(NotImplementedAlert);
                    break;
                case InfoInfoButton:
                    FrmAlert(NotImplementedAlert);
                    break;
            }
            break;
        case popSelectEvent:
            fillInfo();
            break;
    }
    return handled;
}

/*****
 *
 * FUNCTION:    fillInfo
 *
 * DESCRIPTION: Fills in proper info for selected item
 *
 * PARAMETERS:
 *
 * RETURNED:
 *
 *****/
void fillInfo(void) {
    ListType *list = GetObjectPtr(InfoProductList);
    FieldType *brand = GetObjectPtr(InfoBrandField);
    FieldType *size = GetObjectPtr(InfoSizeField);
    FieldType *price = GetObjectPtr(InfoPriceField);
    FieldType *unitprice = GetObjectPtr(InfoUnitPriceField);
    FieldType *alerts = GetObjectPtr(InfoAlertsField);

```



```

char temp[256];

// Update info fields based on the selected product
StrCopy(temp, "");

// First, clear all of the fields
FldDelete(brand, 0, FldGetMaxChars(brand));
FldDelete(size, 0, FldGetMaxChars(size));
FldDelete(unitprice, 0, FldGetMaxChars(size));
FldDelete(price, 0, FldGetMaxChars(price));
FldDelete(alerts, 0, FldGetMaxChars(alerts));

// Now, put the appropriate info in each field
StrCopy(temp, product[LstGetSelection(list)].brand);
FldInsert(brand, temp, StrLen(temp));
StrCopy(temp, product[LstGetSelection(list)].size);
FldInsert(size, temp, StrLen(temp));
StrCopy(temp, product[LstGetSelection(list)].price);
FldInsert(price, temp, StrLen(temp));
StrCopy(temp, product[LstGetSelection(list)].unitprice);
FldInsert(unitprice, temp, StrLen(temp));
// Calculate proper alerts
alertCheck(LstGetSelection(list), temp);
FldInsert(alerts, temp, StrLen(temp));
}

/*****
 *
 * FUNCTION: alertCheck
 *
 * DESCRIPTION: checks ingredients of selected product vs. ingredient
 *              prefs and adds an alert for any banned ingredients
 *
 * PARAMETERS:
 *
 * RETURNED:
 *
 *****/
void alertCheck(Int8 productID, char *alerts) {
    int i;
    Boolean hasAlert = false;
    char temp[50];

    StrCopy(alerts, "");
    //if (ingredientAlertMode) {
        for (i=0; i<NUM_INGREDIENTS; i++) {
            if (product[productID].hasIngredient[i]) {
                if (ingredientPref[i] == BAN_INGR) {
                    hasAlert = true;
                    StrCopy(temp, "Contains ");
                    StrCat(alerts, StrCat(temp, ingredientName[i]));
                    StrCat(alerts, "\n");
                }
            }
        }
    //}
    if (!hasAlert) {
        StrCopy(alerts, "<no alerts>");
    }
}

```

APPENDIX 9 - DATABASE OPERATION

a) Query Optimisation

When a DBMS receives distributed requests requiring data allocated to more than one DBMS, for the same query, query optimisation becomes necessary. It is a process in which the DBMS queried decides how to break down the query into sub-queries, each of which are addressed to a specific DBMS. The query optimiser is crucial to maintaining an effective database of this size and complexity, since it is responsible for

- selection of the order of operations
- selection of methods of execution,
- definition of strategy for execution,
- definition of strategy for data transmission, and
- result allocation,

all of which, if not optimised, would vastly increase the response time of the database. Thus all queries will go through a data optimiser before being answered.

The query optimiser will receive the query in SQL (Structured Query Language), analyse it to identify errors, and check table sizes to produce statistical information. The query will then be translated into internal algebraic form before optimisation begins.

The optimisation process is in three stages: first, all convenient algebraic transformations will be executed; secondly, optimisation will be carried out depending on the cost model and type of data access methods supported, and thirdly, code will be generated to obtain a physical data access program.

The second stage is the most system-specific. The optimiser will decide on the best plan using cost calculations and a decision tree. Cost may be calculated using the following formula:

$$C(\text{total}) = C(i/o) \times n(i/o) + C(\text{cpu}) \times n(\text{cpu}) + C(\text{tr}) \times n(\text{tr})$$

where C denotes quantity of data, n unit cost, (i/o) input and output transactions, (cpu) processing transactions, and (tr) data transmission. The quantity of data transmitted on the network is often a large factor here, and should be minimised where possible. A typical decision tree is shown in *Atzeni* (see references section).

Each node indicates a choice of a particular option (with regard to order of operations, result allocation etc.). Thus each leaf node represents a specific execution plan. The optimiser will search for the leaf node with the lowest corresponding cost, using the above formula to evaluate each operation that appears as a node of the tree along the path to that node.

Since the user profiles will be constantly altered, queries will be of the 'compile and go' type, ie used once but not stored for re-use. Consequently, the optimiser will be satisfied with a near-optimal solution, rather than spending extra time searching for the optimal solution when the query will only be used once.

Thus a strategy for answering distributed queries will be developed, and response time will be greatly reduced.

Parallelism: Database queries are repetitive by nature. Thus computations lend themselves to being carried out in parallel very efficiently. A complete scan carried out using n scans each on a portion of a database, if stored on n discs managed by n processors, will have a response time equal to 1/n of the time required for the same scan carried out in series.

For this reason, where possible this database will use inter-query parallelism, whereby different queries are carried out in parallel (as opposed to different parts of the same query). This is appropriate here, since the queries will be simple searches or sums, not complex, time-consuming calculations, and since queries will arrive at a high frequency due to the use of the Internet.

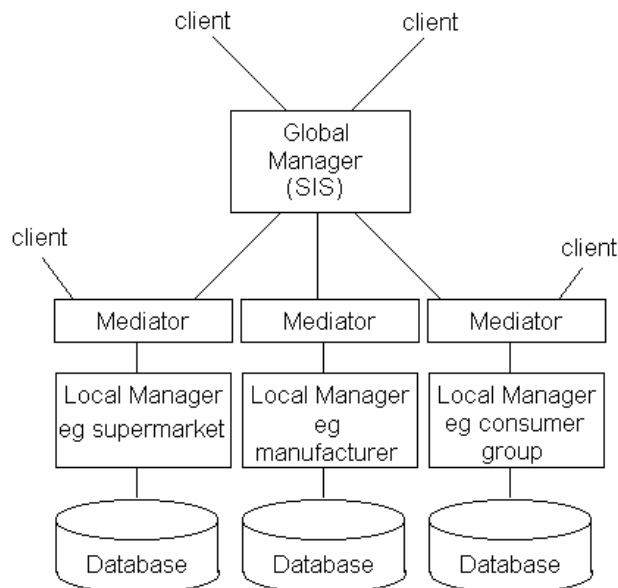
Making use of parallelism enables each processor to be allocated part of the total load. The database will possess a **dispatcher** whose sole purpose is to redirect queries to appropriate servers according to

a strategy. Typical strategies include equalising the load on each server, and sending each query to the server that has the most efficient access to the data involved. More work will be needed to determine the optimal strategy.

b) Co-operation Between Existing Databases

For the DBMS to be compatible with the existing databases, **interoperability** and **co-operation** must be ensured. Interoperability is process-centred, and is the capacity for interaction, which requires adaptability and conversion functions which make possible the exchange of information between system, networks and applications even when heterogeneous. Co-operation is data-centred, and is the capacity of an application system to make use of application services made available by other systems and managed by different organisations.

Interoperability: This is made possible using standard protocols to carry out standard tasks. These protocols guarantee compatible architecture between heterogeneous databases such as ours. The database will comply with *Open Database Connectivity*, a restricted SQL language that guarantees access to remote data, and *X-Open Distributed Transaction Processing*, an interface which guarantees interoperability of transactional computations between DBMSs.



Co-operation: This property is more data centred. The data here is autonomous, heterogeneous, and widely distributed, all of which are problematic for co-operation between databases. In particular here, there is a need for high levels of **transparency**, **complexity** and **currency**, which are explained below. A proposed solution is shown in the diagram.

The architecture shown solves the problem of co-operation using a multi-database system. Each external database continues to be used by its local user (ie stores or manufacturers) while also being accessible by modules called **mediators**. The mediators **transform and filter the accesses**, showing only the required/permitted portion of the external database, to a global manager the Smart Information System central database).

Transparency: The presence of the global manager presents an integrated view to the user, who sees the database as if it were a single entity, ie a high level of transparency. This is important as the user is the general public, who cannot be expected to deal with discrepancies between databases.

Complexity: Due to the presence of the mediators, data cannot be modified by the global manager. Thus, the database has a high degree of complexity. This is vital, since the stores, manufacturers, etc. will only be prepared to share their information if they are confident that it will be secure.

Currency: Data is accessed directly, without the presence of a data warehouse to store often-used data, and so the currency of data is high. This is especially important, as prices and ingredients of products change frequently.

APPENDIX 10 – LETTER TO POTENTIAL INFORMATION SOURCE



Oriel College, Oxford, OX1 4EW
Tel: 01865 276555

12th November, 2001

Dear _____,

I am part of a postgraduate global research team, based in Oxford University, which is designing a product that may in the future be of interest to your company. The product is currently at a very early stage in its conception, and as one of the UK's largest supermarkets, I would like to ask for your opinion on one aspect.

The product is called 'Aware'. Its purpose is to enable customers to make sense of the vast amount of information available to them regarding the many choices that they make between products when they shop.

It is a portable device, which may be store owned or personally owned, incorporating a wireless Internet connection and a scanner. The scanner is used to scan the barcode of a product, which will be used to identify the product and access a wealth of information about it on a central database, via the Internet.

The user will have pre-set the device to their preferred 'user profile' which will tell the device what information they do and do not want. For example, a user may specify that out of all the available information, they only wish to be told whether or not the item contains meat, how much fat it contains, its price, and whether it is tested on animals.

There are also functions which enable the user to make a shopping list at home, and many other time-saving features.

The aim of the product is to pare down the vast amount of information customers deal with in supermarkets, and to give customers confidence in their choices when faced with a huge array of similar products. Consumers will, quite simply, be more confident that they are buying what they want, and will therefore be more inclined to buy.

This product will obviously require a huge database to be constructed. As a large UK supermarket, you will of course have your own database of such information (the vast majority of which is contained in the food label). Access to this information would be the most direct and efficient way of constructing our own database. The information would be securely held, and combined with that from other supermarkets and consumer groups, to give an unbiased platform from which customer queries could be answered using our device.

I remind you that this is merely a postgraduate research/design project, and that agreeing to share your information in theory does not in any way commit you in practise! However, we would welcome an honest response, as well as your views on the usefulness of such a product.

Many thanks,
Yours sincerely,

Sara Donner,
Oriel College, Oxford University.

APPENDIX 11 - INFORMATION SOURCES

	UK	USA	Korea	Venezuela
Major Supermarkets	Tesco Sainsbury Safeway Asda		E-Mart Magnet Carrefour Kim's Club	Makro Cada Excelsior Gamma Central Madeirance
Regulatory and Consumer Information Groups	Institute of Grocery Distribution Foodsector.com	US Food & Drug Administration		

APPENDIX 12 - SAMPLE REPLY FROM POTENTIAL INFORMATION SOURCE

Sara Donner,

As information that may be pertinent to your project, the Food and Drug Administration (FDA) does not maintain a database of the formulas or ingredients food products manufactured in the United States (U.S.) or manufactured in other countries. Also, FDA does not register the manufacturers of most categories of food. My office occasionally receives inquiries similar to yours from persons who intend to develop databases of all or most manufactured food products.

A problem with this idea is that food manufacturers may occasionally or even frequently change their formulas due to availability of certain ingredients or changing comparative prices of ingredients. As a result of such ingredient changes, the database would quickly become inaccurate. You may wish to check with your legal advisors to determine if your company or institution would incur legal liability in the event of injury or death of a consumer(s) due to inaccurate information in your database. As an example, consumers who have peanut allergy may die from asthmatic attack if a manufacturer changes shortening ingredients in a bakery product, substituting peanut oil for another shortening ingredient. FDA's webpage "Information about Food Allergens" is on webpage <http://www.cfsan.fda.gov/~dms/wh-alrgy.html>.

LABEL ACCURACY During FDA's inspections of food factories in the U.S. the agency routinely checks ingredient statements on labels and obtains correction of inaccurate labels. Recent recalls of foods due to inaccurate labels or other reasons are listed on FDA's webpage "Recalls and Safety Alerts" <http://www.fda.gov/opacom/7alerts.html>. Recalls are actions taken by a firm to remove a product from the market. Recalls may be conducted on a firm's own initiative, by FDA request, or by FDA order under statutory authority. A Class I recall is a situation in which there is a reasonable probability that the use of or exposure to a violative product will cause serious adverse health consequences or death. A Class II recall is a situation in which use of or exposure to a violative product may cause temporary or medically reversible adverse health consequences or where the probability of serious adverse health consequences is remote. A Class III recall is a situation in which use of or exposure to a violative product is not likely to cause adverse health consequences.

MEAT PRODUCTS Another government agency, the United States Department of Agriculture (USDA) has jurisdiction over meat and poultry products that are manufactured in this country or imported. The address of the USDA Homepage is <http://www.usda.gov/>. You may wish to contact USDA at: Technical Service Center Food Safety Inspection Service U.S. Department of Agriculture Omaha, Nebraska Telephone 1-800-233-3935, 1-402-221-7488, 1-402-221-7400 E-mail FSIS.Regulations@usda.gov

The Food and Drug Administration (FDA) regulates non-meat foods. You may wish to send any further inquiries concerning FDA regulations for foods or cosmetics directly to Industry Activities Staff at oco3@cfsan.fda.gov or telephone our staff at 1-202-205-5251 (Washington DC-hours 7:00 am-4:30 pm). Depending on staff availability, we may be unable to respond to some e-mail inquiries during periods when a large number of requests are received.

You may also wish to call our toll-free number 1-888-723-3366 to hear informative recorded messages. /s/ Industry Activities Staff Center for Food Safety and Applied Nutrition Food and Drug Administration E-mail oco3@cfsan.fda.gov Telephone 1-202-205-5251

APPENDIX 13 – SQL CODE FOR DATABASE QUERIES

1.Single Scan Query

```
SELECT [Product ID].Barcode, [Product ID].[Trade Name], [Product ID].[Descriptive Name],  
[Product ID].Manufacturer, [Product ID].[Weight / Volume], [Product ID].Nuts  
FROM [Product ID] INNER JOIN Store ON [Product ID].Barcode = Store.Barcode  
WHERE ((([Product ID].Barcode)=[Type barcode here, including spaces:]));
```

2.Compare Query

```
SELECT [Product ID].Barcode, [Product ID].[Trade Name], [Product ID].[Descriptive Name],  
[Product ID].[Weight / Volume], Store.[Tesco Price], Store.[Sainsbury Price]  
FROM [Product ID] INNER JOIN Store ON [Product ID].Barcode = Store.Barcode  
WHERE (([Product ID].[Keyword])=  
    (  
        SELECT ([Product ID].[Keyword])  
        FROM [Product ID]  
        WHERE [Product ID].Barcode = [Product ID].[Type Barcode Here]));
```


APPENDIX 14 – OBSERVATION OF SHOPPERS

OBSERVATION OF SHOPPERS AT BUSCH’S GROCERY IN ANN ARBOR

Three types of shoppers:

1. **Rushed/Efficient** – Makes decision within 5 seconds. Probably buys mostly familiar brands. Scans quickly for lower prices, but relies mostly on intuition in the interest of speed. Walks deliberately from one item to the next, ignoring sale displays.
2. **Quick Look** – Takes the time to look carefully at different options before deciding. Sometimes looking at every side of the box, but not taking enough time to read anything significant. May swap between several brands before deciding on one. Probably relies on intuition and impression based on product packaging. Gazes across products to see if something catches the eye.
3. **Careful** – Takes the time to actually read the ingredient, fat content, etc and compare prices before deciding on a purchase.

Three types of transport:

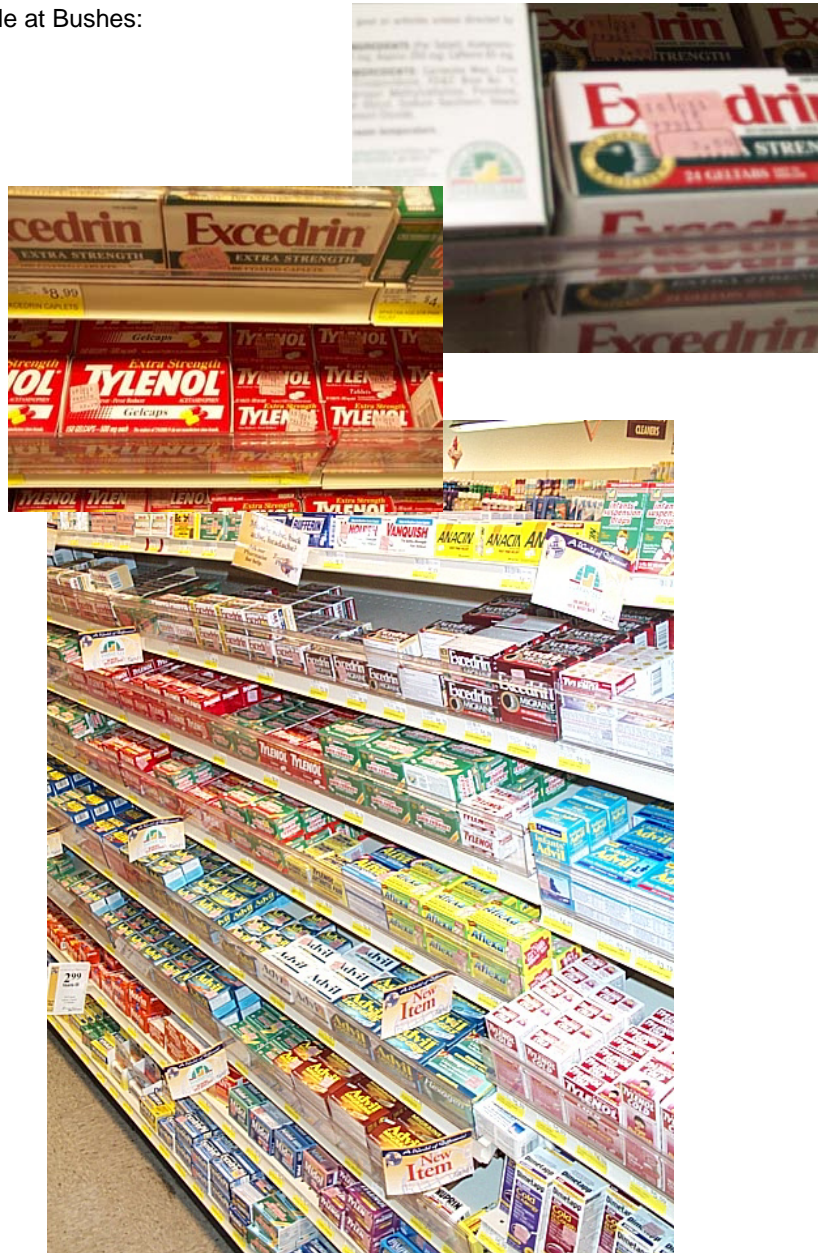
1. **Basket:** for longer shopping trips
2. **Hand Basket:** for a quick trip to get a few things
3. **No Basket:** especially for rushed/efficient shopper.

Notes:

- Checking shopping list from pocket: doesn’t carry around shopping list. Instead, returns list to pocket when not being used.
- Spending time examining different products also seems to be part of the thought process. Shoppers use the time to think “do I need this?” or “what should I make for dinner?”. This thought process is easier when the physical product is right in front of you.
- Many shoppers pick up items, examine, and return without purchase. Many times actually physically picking up 2 to 4 products before deciding.
- Some young children go along quietly for the ride, while older kids tend to bug their parents to buy things
- Food bar and deli in store – pre-prepared food is a big thing. Convenience.
- Sales displays placed in the middle of aisles to attract shoppers.
- One customer spent a few minutes pacing back and forth around the meat section. He was looking for lamb, but couldn’t find it. Finally, he went to the deli and got a store worker to help him locate the lamb. The worker said that they move things around once in a while for organization.
- **Idea:** have a digitized shopping list as part of your Palm Pilot. Then, when you walk into the store, the items that you need can become highlighted on a map so that you know where the items you want are located.
- Elder shoppers are slow moving and quite picky. They argue over prices and coupons.
- Spoke to woman who gives out free food samples. She works 5 days per week, and her salary is paid for by the increase in sales due to advertising them in the aisle.
- Child wants sugar cereal, but mom insists on better cereal. But then she buys Life cereal, which is loaded in sugar. I don’t think she realizes. The box looks more like a healthy cereal box, but the cereal is actually very very sweet.
- Some shoppers spend a long time staring at their choices, but all they see is the front label graphic design.
- Checkout line includes items such as newspapers, folding toothbrushes, chapstick, gum, sewing kit, comb/brush, pain relievers, and candy. These are the kind of items people are likely to buy if they see them while waiting. Why these items?
- People don’t seem to bother counting change or checking credit-card receipts after checkout. They seem to have a lot of trust in the system in the interest of convenience and efficiency.

Some of the pain relievers available at Bushes:

- Bayer
 - Original strength
 - Genuine
 - 50 / 100 capsels
 - Extra strength
 - Safety coated
 - PM
- Spartin
 - Coated aspirin
 - Bottle or box
 - For infants
 - Ibuprofin
 - Added strength
 - Extra strength
 - Stay awake
 - Naproxen Sodium
- St. Josephs
 - Chewable
 - Coated
- Brite-Life
- Ecotric
 - For arthritis
- Bufferin
- Vanquish
- Anacin
- Tylenol
 - Liquid vs. solid
 - Extra strength
 - Woman's
 - PM
 - Arthritis pain
- Advil
 - Caplets
 - Tablets
 - 24 / 50 / 100 pills
- Motrin
 - 24 / 50 / 100 pills
 - Migrane
- Vivarin
- Excedrin
 - Extra strength
 - PM
- Advil
 - 24 / 50 / 100
 - Migrane
 - Liqui-gells
- Pamprin
- Dramamine
- Aleve
 - All day strong



Note: all pain relievers are

- aspirin
- acetaminophin
- ibuprofin
- caffene
- naproxen sodium

or some combination. (Some of the PM medicine also contains diphenhydramine citrate for sleep). Yet, there are at least 17 brands here, and over 50 different packages to choose from. This is just pain relievers – no cold medicine or products for other symptoms. It took me 15 minutes just to write down some of the different options available. How long would it take to really compare brand ingredience and make an informed purchasing decision?

There are so many options to choose from. How do you know which one is best for you?

Wines



Insect poison



Condoms



Greeting Cards



Spaghetti sauce



Salad dressing



Even more salad dressing!



Signs attract people to certain items:



Shoppers make choices on every product they purchase:



Is the cheaper brand the better deal?



Tools for shopping:



Grocery stores have large deli sections. Americans like pre-prepared and frozen foods because it saves time in cooking.



Loads of information and loads of options.

Good stores have people available to help customers with questions. Some stores have phones in the aisles to page for help if you can't find anyone.



When customers have to fill out information, they often have to fill out the same information many times:



APPENDIX 15 – SYMBOL CSM-150 SCANNER MODULE SPEC SHEET