Baby Xanadu, an E-Commerce Site

By

Steve Sharpshair

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

May 2001

Baby Xanadu, an E-Commerce Site

by

Steven D. Sharpshair

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

_____	_____
Steven D. Sharpshair	Date
_____	_____
Robert Schlemmer, Faculty Advisor	Date
_____	_____
Lawrence G. Gilligan, Department Head	Date

Table of Contents

List of Figures

Abstract

A new baby in the family can be a joyous experience for everyone involved, bringing out friends and relatives for parties, religious ceremonies and general well wishing. Giving baby toys, clothes and baby accessories has become an integral part of almost every new baby celebration. All these gifts have to come from somewhere, and with Internet retail becoming more popular, the idea of a profitable online baby store becomes plausible. This paper explores the creation and attributes of *Baby Xanadu*, an online baby supply store created as my senior design project. Baby Xanadu has all the functionality of an e-commerce site. It is dynamically created using Active Server Pages and a Microsoft SQL Server database and has been built to be browser independent, though cookies are required. Customers can browse through the various products on the site, put them in a shopping cart and go through a checkout process. Customers can also set up a gift registry where they can add as many products as they desire. Any shopper can search for a particular registry, purchase gifts from it and have it sent directly to the registry owner. This project required a lot of research and coding, which can be hard to express in writing, this paper will attempt to explain it all to the reader's satisfaction.

Baby Xanadu, an E-Commerce Site

1. Statement of the Problem

It is required in Senior Design to choose a project utilizing the knowledge that we have learned over the course of our studies. Deciding exactly what project to do can be a hard choice. Some students may use real life projects, either for their jobs or other sources, but students who don't have this option must decide on their own what exactly they want to do, and how much effort are they willing to put into it. I wanted to choose a project that would relate to my career goals.

1.1 Definition of Need

Why build an online baby store or any E-commerce site for that matter? According to an E-merchant study released earlier last year by Internet market research firm Keenan Vision, the number of e-merchants will number 400,000 in 2003 (4, p. 1). The third quarter of last year witnessed a 15.3 percent online sales gain over the previous quarter (4, p. 1). E-commerce is here to stay, as time goes on more E-commerce sites are being built, which means that competition increases. This will be somewhat offset by the increase in Internet shoppers, but not necessarily. Establishing a nicely constructed and easy to use E-commerce site now, could create loyal shoppers who would continue to use the site in the future, even though similar sites may, by then, be around. Going through the process of researching and developing an E-commerce site will be good experience for someone who wants to establish a career in the field.

2. Review of Literature

E-commerce has become a high profile and exciting part of modern business. There are numerous articles and books dedicated to the subject. For the development of this project, four instructional books and three websites along with my own experience were the primary resources. These cover a categories from Microsoft's SQL Server database creation, programming in Active Server Pages (ASP) and general E-commerce dos and don'ts.

2.1 Book Resources

As mentioned before there were four primary books that I used in the development of this project. They are:

- *Using Microsoft SQL Server 7.0* by Stephen Wynkoop
- *Beginning E-commerce with Visual Basic, ASP, SQL Server 7.0 and MTS* by Mathew Reynolds
- *Sams Teach Yourself E-commerce Programming with ASP in 21 Days* by Stephen Walther, Jonathan Levine
- *Practical Visual InterDev 6* by Michael Amundsen

A database is the backbone of this project; the technical information on how to create the various tables, views and stored procedures was primarily supplied by the book *Using Microsoft SQL Server 7.0*. This book helped a lot during the creation of SQL select, delete and update statements used within the project's views, stored procedures and VBScripts. The book itself teaches from database basics to advanced SQL Server attributes and techniques.

In order help discover the elements needed within an E-commerce site two books were often used. *Sams Teach Yourself E-commerce Programming with ASP in 21 Days* and *Beginning E-commerce with Visual Basic, ASP, SQL Server 7.0 and MTS* were very helpful in showing examples and methods for the creation of an E-commerce site. Both books stepped through the process of building a site, yet both took different approaches and gave information pertaining to that approach. The *Sams* book concentrated primarily on ASP creation using VBScript for server side scripting, while the other book melded Visual Basic, ASP, SQL Server and MTS together to create a very robust site. Both books were great help and I would recommend them to others.

In order to help facilitate the creation of this site, Microsoft's Visual Interdev 6.0 was used. Visual Interdev is broad ranging Web site development software that combines HTML objects, database connectivity, script creation, a color-based editor and FTP capabilities into a very nice ASP development tool. In order to learn the use of this product the book *Practical Visual InterDev* was used. This book goes through the basics of HTML, ASP, VBScript and Visual Interdev. Teaching them through examples and good graphics. The book even shows how to use basic ADO code for database connectivity. The book was very helpful

2.2 Internet Resources

The Internet is full of information on almost any subject imaginable, especially on Web programming and E-commerce. Three Internet sites were heavily used during the creation of this project.

They are:

- 15 Seconds, www.15seconds.com
- DevGuru, www.devguru.com
- E-Commerce Times, www.ecommercetimes.com

15 Seconds is a great site for Internet specific programmers, covering subjects on ASP, database, scripting, HTML, XML and more. The site has articles showing specific ways to code for various Web site development issues with a nice search engine to find them. It also has list servers, which allow programmers to communicate questions, ideas and philosophies about development. *15 seconds* was browsed quite regularly during the development of this project.

DevGuru was also used heavily as a quick reference for ASP and VBScript definitions and examples. The site has over three thousand pages containing comprehensive quick reference guides, tutorials, knowledge base articles, and useful products to serve a wide range of developers' needs. This site came in very handy, its quick and easy to used menus made finding information quicker than looking it up in a book.

Coding was just a part of this project, although, a major part, but before coding could begin, an idea on what others believe are important to an E-commerce site was needed. The *E-Commerce Times* has up-to-date articles on E-commerce issues and news. During the initial researching for this project, *the E-Commerce Times* provided some good information on subjects like statistics on Internet usage and E-commerce site needs.

3. Description of the Solution

3.1 User Profile

The intended user for *Baby Xanadu* will be someone comfortable using the Internet to shop and purchase items. Since the site's main product will be baby-related items, customers will consist of soon to be parents, parents of babies or friends and relatives of a baby's parents. The site will have an online registry and will ship an order anywhere in the U.S., so out of town friends and relatives can easily purchase gifts and have them sent to the registered parents.

3.2 Design Protocols

3.2.1 **Site Organization** see Figure 1.

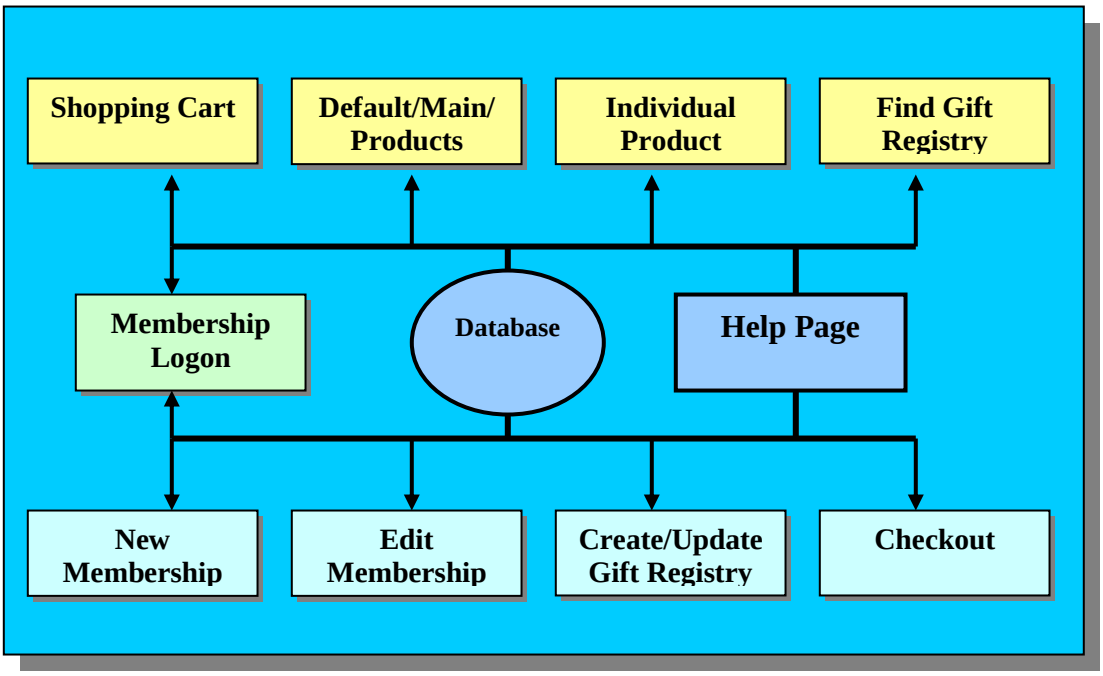


Figure 1. Navigation Schematic

3.2.2 Navigation

The main page for the *Baby Xanadu* Web site is the default page. This default page will initially display a header on top with all the main site links attached to it. The site links will consist of Product Categories, User Account, Shopping Cart, Help, Gift Registry, Main Page and Checkout. Located below the header and to the left will be a navigation section of dynamic links displaying each main product category and the categories immediately beneath them. There are six main product categories: Mothers; Nursery; Gear; Care, Health and Feeding; Baby Clothing; Toys, Books and More.

All links to the product categories lead back to the default page, but with additional criteria. Using these criteria the Web server can determine from the page code what is to be displayed. If a product category link is followed from the header or left navigation section, then the page will display the entire selected category's child categories and their child categories, if any. To the right, a path link list will be displayed, showing the current followed category path; also the product specials will change in accordance to the selected category.

Eventually a category will have no child or sub categories, when this occurs the display of all of the products within that category will be displayed and the left navigation section will vanish. Initially the product page will display only ten products at a time, with a forward, backward and page links. These product page navigation links actually will call on the same product page, but with a page criteria being sent. Every product will display a picture of the product, its name and manufacturer, a brief description and its price. It will also link to a product details page, where the details of the product and the ability to add it to the shopping cart are displayed.

The product detail page will have the same header and header links as the default page, but again, like the product display page, there will be no left side category navigation links. The product link path will also still displayed. A larger picture of the product will be displayed, as well as a more detailed product description and the price. To the right is a list box containing all the variations of the product, like color and sizes, and whether the product for that variation is in stock. Images of each available product color will also shown below the product picture. Below the list box will be a text box holding the quantity of the product the customer wants to add to their shopping cart. The default quantity will be one, but the customer can change that. Below the quantity box will be a button that can be used to add the product to the shopping cart. A link back to the product list is also displayed.

If the product is added to the shopping cart, then the shopping cart will be displayed, listing all of the products and their quantities currently in the cart. The shopping cart has the same header and header links as all the other pages. Customers can also edit their shopping cart from the shopping cart page. They can change the quantities of the product or remove the product from the cart. There will be navigation links back to the page where the cart was originally called. There will also be a button leading to the checkout, thus two links to the checkout will exist, one by the button and one in the header links.

If the customer chooses to checkout, the user is first routed to the members' sign in page, where they can enter their username and password to logon. If they are not already a member, then they are directed to the new members' page where they can sign up by giving their name, billing address, telephone number, Email address (their

username) and a membership password. They are also able to enter credit card information to be used in their orders, but which is optional. Once they have entered the required information, and it is checked and added to the database, the membership page is again displayed and they can log in. Once they are logged in, they can then proceed to check out.

During the first step of check out the customer will enter where they want the order to be shipped. They can send either to their billing address located in their profile, a separate address, or if they shopped from a gift registry (which will be discussed later), the address of that registry. Once the address is entered they can select "Continue" and proceed to the second step where credit card information must be entered and verified. The name and last five digits of any previous credit card used by the member will be displayed. The customer can choose one of these cards or enter new credit card information, then click next. The credit card information would normally then be checked for validity, but that is beyond the scope of this project, because of the fees involved. When the credit card information is chosen and the "Continue" button is selected, the products being purchased, their price total and shipping choices and costs are displayed. There will be three shipping choices: next day, first class, or ground. Selecting the "Continue" button will load the confirmation page where the billing address, mailing address, product and price information, shipping information and credit card information are displayed. After the customer verifies this information, they can then process the order by clicking on the "verified" button. Once verified the customer is given an order number that can be used to track their order while it's still in house. An order confirmation email will also be sent.

A member is also allowed to set up a gift registry. The process of adding products to registry uses the same pages as adding products to the shopping cart, but they must first enter registry mode. A registry link will be made available within the default page. When selected a page is loaded with the options to create a registry, edit a registry or search for a specific person's registry. If create registry is chosen, then the customer is directed to log in or create a login from the membership pages. Once this is accomplished they enter registry mode noted by a change it the header image with a color change and the appearance of view registry and exit registry buttons. Once in registry mode customers can add products to their registry. They will actually be using the same default, product and shopping cart pages as a normal shopper, but instead of using the shopping cart database, they use the registry database. Once they have finished they will be able to exit the registry mode by using an exit registry button that will always be available while in that mode. They can later edit and view their registry. If a customer wants to purchase products off a registry, they must perform a search from the registry page by state and baby or parent name. Once they have found and chosen the desired registry, they will be put in registry purchase mode. Any products they add to the shopping cart are designated for the registry owners. When they decide to checkout the order, they are given the option of sending the product directly to the registry owner, without actually displaying the address. A product will be removed from the specific registry once its order has been successfully processed.

At anytime a customer will be able to access help from the main header links, which are displayed on every page.

3.2.3 Colors and Graphics

There will be five colors that will be associated with the site: white, purple, green, blue and yellow. The background color for most of the site will be white. Purple is used in most of the header image. A pale green will be used as a background in the left navigation section of the default page. Most text will be black, except for a few places where red is used to help the text stand out.

Graphics will be primarily be used in the header, including the title “Baby Xanadu.” . Navigation buttons will be images instead of the basic gray button default. The buttons images will still be text, but will go with the color scheme.

4. Project Objectives

4.1 Easy Interface

The interface will be intuitive and easy to use and understand. Menus and navigation elements will be noticeable and easy to figure out. A shopper will be able to glance at a page and get all of the relevant information they need to navigate through the site. Searching for a button that doesn't appear to be a button is not appropriate for a retail site.

4.2 Product Descriptions/Information

Products will be divided into categories. Each category may have a subcategory and so on. Each product will have a picture and a description associated with it along with color, sizes, manufacturer, price and sale price (if any). All this information will be stored in the database in two tables with a one-to-many relationship. The first table will hold general product information like manufacturer, names, price and a general product ID, while the second table will hold product details, such as size, color and product ID. This will allow a product, such as a shirt, that has many different sizes or colors to be grouped together with the same style shirt but a different size or color.

4.3 Shopping Cart

A shopping cart has become a necessity for online shopping. This allows the shopper to add items they are interested in purchasing to their cart, which will be tallied when the shopper is finished shopping. The shopper will be able to view and edit his or her shopping cart at anytime. If they have changed their mind about an item, they can easily remove it from their shopping cart.

A cookie is used so whenever a customer visits the site, their computer is checked for it; if the cookie exists, the customer's previous shopping cart is loaded. This give the customer the ability to keep their shopping cart longer than twenty minutes of inactivity, the normal session default, before being dropped.

The shopping cart will be accessible from anywhere in the site. There is a link to it in the top right of the page header along with a total of the amount of products and their total cost that have been placed in the shopping cart.

4.4 Membership

In order to purchase items a customer must become a member of the site, which is free. There will be a membership link within the page header located on every page. From the membership page a customer will be able to log on or create a new account. If they create a new account they will be sent to a forms page where they will need to enter their name, address, phone number, Email address and an account password.

4.5 Checkout

When the shopper is finished shopping they will move to the checkout either from the header link, or the link in the shopping cart. In actuality the customer will first be sent to the membership page to log in or create a new account then sent to the checkout page. Once at the checkout their items will be tallied, shipping will be chosen and costs will be added, sending address information will be gathered, and payment will performed. This will take place over a few pages and the shopper can cancel at anytime up until they click on the final verify purchase button. Once they select the "Confirm Order" button they will be given an order ID and an email of their order will be sent to them.

4.6 Gift Registry

The ability to register gifts online will made available. To create a gift registry, a member will have to first log in from the members page, then click on create registry from the options presented. They will then be in registry mode and be able to add orders to the registry just like they would the shopping cart. Once the customer has finished choosing items, they can log out of registry mode from a convenient button. Shoppers will be able to purchase registry items by clicking on the registry link and performing a search based on either the father's, mother's or baby's name. They will then see a list of the items in the registry and the quantity desired. They can add them to their cart and during checkout, send the products to the registry's owner.

5 Proof of Design

To prove that the project's design corresponds to the objectives of the project, each objective will be compared to the final result along with how it was accomplished.

5.1 Easy Interface and Navigation

The site's interface was designed to be easy to use by its customer's. Bright colors have been used, primarily purple, greens, blues and yellows. Navigation through the site is simple and can be accomplished through the header, which appears on every page. The header has a link to the shopping cart located in the upper right corner, which takes the user to the shopping cart page, allowing them to view what they have already put in for purchase. The header also has links to each of the six main shopping categories, set up to look like a toy train (see Figure 2). Each car image in the train resides in a cell of an HTML borderless table and is linked to the default page, passing on the category ID of the specific category by using a query string. The header page also has links to the main page, gift registry, checkout, account information and help.

Figure 2. The header that is shown on every page.

Navigation within the dynamic default page, *default.asp*, where products and product specials are displayed can be accomplished by using the left navigation section, easily identified with a light green background and links to the various product categories (see Figure 3). These links are built dynamically in a type of tree structure, where each category has a parent and a child category, except for the categories at the top and bottom

of the tree. The links are created using server side VBScripts and an application array that holds all of the categories' IDs to build the query string, holding the category ID, for each link. The application array is created once from the database using an ADO recordset and the *getrows* method. Stored in memory, the application array allows quicker access time for category information than a database. If the application array is missing do to server problems, the array will be rebuilt automatically.

Figure 3. The main page showing the various links available and product specials.

There is also one other way to navigate through the site, every page that displays product information, except for the main page, also displays a category path of links, to help the customer know where they are, and to help them backtrack.

5.2 Product Descriptions/Information

An E-commerce site's whole purpose is to sell items. Presenting items to the sites customers in a memorable, visual and informative way is a must. This project uses ASP, VBScript, ADO and the category application array to show a picture of the product, a brief description of it and its price. Four product specials are displayed at each level of the category tree; specials for the current category and its subcategories are chosen randomly from the database at each level (see Figure 3). Any product that has a sale value is considered a special. Products, ten to a page, from a specific category are also displayed when the bottom of the category tree is reached (see Figure 4). For instance "tops" is the bottom category for "girls baby clothing", once "tops" is chosen form the left column links, a list of baby girl's tops is displayed. When a category's products are displayed, the left navigation column of links vanishes, and just the items are show. Both the specials and the category product descriptions are created within the same page, *default.asp.*

Each product has a link to *product.asp* that displays the product in more detail. *Product.asp* pulls in all of the products data from the database using ASP, VBScript and ADO code. This information is located in two tables named *Products* and *ProductDetails*. A bigger picture of the product is displayed, along with more details, the price and sale price, if any, and small pictures of the various colors and patterns that may be available (see Figure 5). Additionally a list box of the various sizes and colors available are located to the right, along with a quantity text box and a button that will add the item to the shopping cart. The list box, quantity box and button are all part of a form, which passes the information to cart.asp, the shopping cart, when the button is selected.

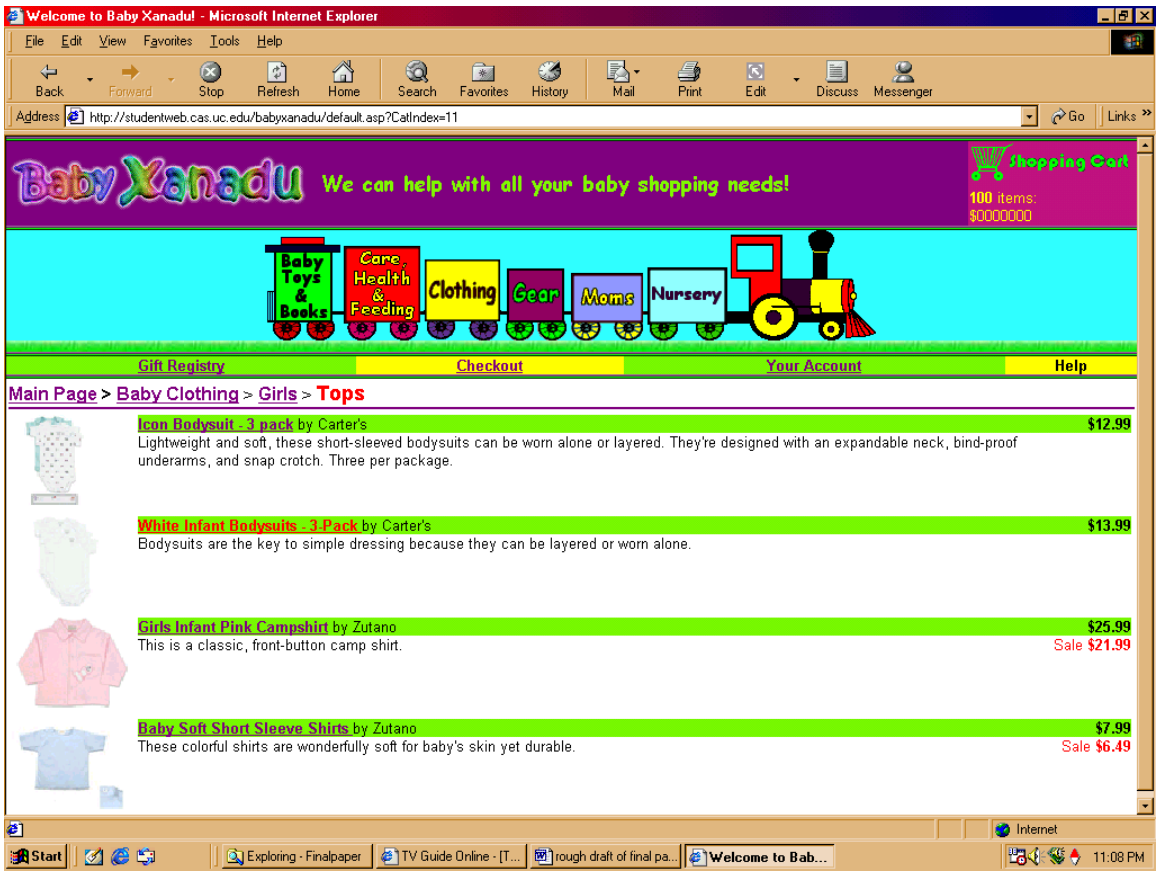


Figure 4. Screen shot of a category's products being displayed in default.asp.

A back button is also made available which will send the customer back to their previous page. This is accomplished by using variables that have been passed in by the previous page via query string, specifically the category ID of the product, the products ID and the page number it was located on.

5.3 Shopping Cart

Items added to the shopping cart are passed in either through *product.asp* or *view_registry.asp* (discussed in section 4.5) using forms. The product's product ID and the quantity desired, along with navigation information for a "Continue Shopping" button are checked for. The session variables "CartID" and "CartTotal" are called on, but if

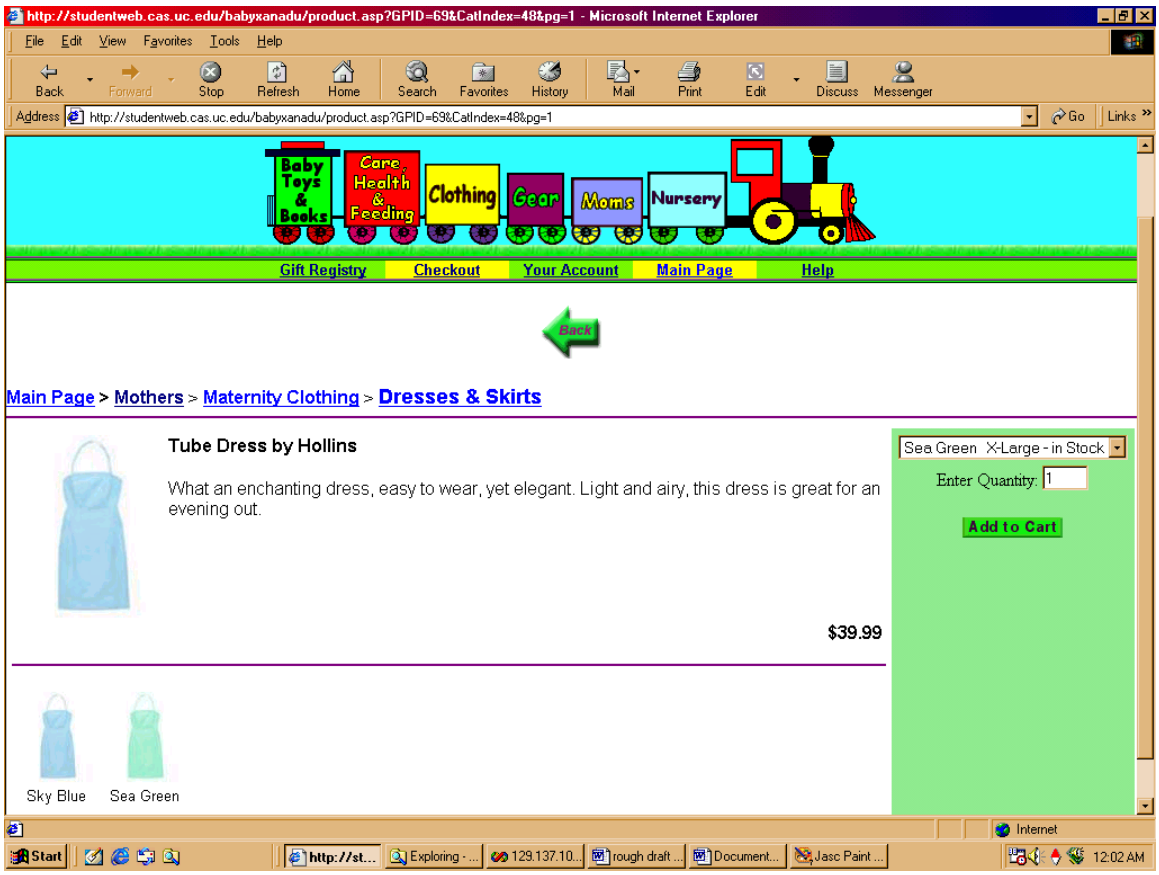


Figure 5. Product.asp displays product information

they do not exist then there is no cart currently assigned to the user, so one must be created. Using ADO code, a new row is created in the *cart* table consisting of the cart ID and an expiration date of two days in the future. The cart ID from the table is then stored as a session variable and put into a cookie, along with the date, and stored on the user's computer. Whenever a person browses to *Baby Xanadu* the global.asa file will automatically try to read the cookie and assign a session variable to the cart ID. The expiration date tells the users computer how long to hold onto the cookie, in this case it is two days. The product is then added to the *CartDetails* table that holds the product ID, cart ID and the quantity of the product to be purchased. In the case where the user is in registry mode, which is checked by looking for the session variable "Registry", then the

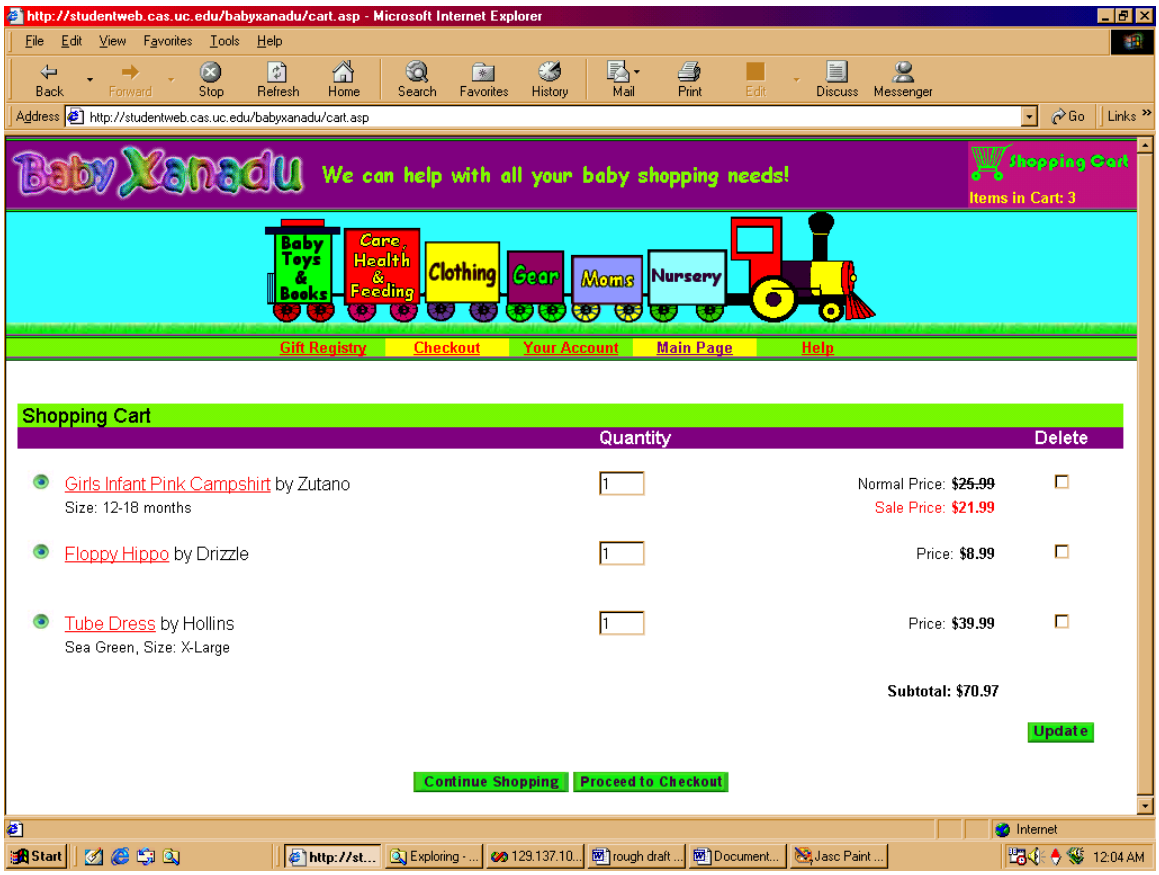


Figure 6. Cart.asp adds new products to database and displays products in cart.

product instead is added to the *RegistryDetails* table, which holds the customer's ID, the products ID and the quantity.

After the product has been added to the database, or the shopping cart has been navigated to from the link in the header, *cart.asp* displays all of the products currently in the shopping cart (see Figure 6). Using ADO and a select statement to search the *CartDetails* table for rows with the current cart's ID and then retrieving the product information from each row from the *Products* and *ProductDetails* tables, a recordset is created. By using a *Do While loop*, the recordset is then moved through, and each row of product information is displayed. Also displayed are a text box holding the quantity of the product to be purchased and a checkbox, and on the bottom a button labeled "Update". If the checkbox is checked or the quantity changed and the "Update" button is

selected, then any items to be deleted will be removed from the cart, and any changes to the quantities will also be changed.

A “Continue Shopping” and a “Proceed to Checkout” button will also be displayed at the bottom of the list. Both buttons submit separate forms, each pointing to a different page. The “Continue Shopping” button uses the navigation information like product ID, category ID and page to determine where the shopper was last, then send them there. The “Proceed to Checkout” button will link to the login.asp page and pass on the value form value “Checkout” to let the login page know that after logging in, the checkout process should begin.

5.4 Gift Registry

The gift registry can be assessed from the header link “Gift Registry”, which points to *registry.asp*. *Registry.asp* has links for either creating or updating a registry and searching for a registry. Selecting the “Create/Update Registry” link will bring up the login.asp page, where the user must either login or create an account. Passed on to the login page is the query string “Mode” with the value of “Registry”, which lets the login page know to proceed to the *edit_registry.asp* page next. In *edit_registry.asp* users who already have a registry can update it and add more items to it. Users who don’t have a registry can create a new one by filling in the fields. The fields include the mother, father and baby’s name along with an address to send any items purchased from the registry. When “Continue” is selected, *edit_registry.asp* processes the form data and if it valid, adds it to the *Registry* table then shows the data again and allows items to be added, if not it redisplays the page and asks for corrections to be made.

Once a registry is created and the user selects “Add Item”, registry mode is implemented and the look of the header changes on its right. “View Registry” and “Exit Registry” buttons appear in place of the shopping cart link, and the background surrounding it turns yellow. Additionally a session variable “Registry” is created that is used by *cart.asp* (see Figure 7). The user can then shop as normal, but items are being sent to the table *RegistryDetails* instead of *CartDetails*. When they view the registry or add items the page called is actually *cart.asp*, but using a series of “IF” statements against the session variable “Registry”, it can decide whether to show the registry or the normal cart.

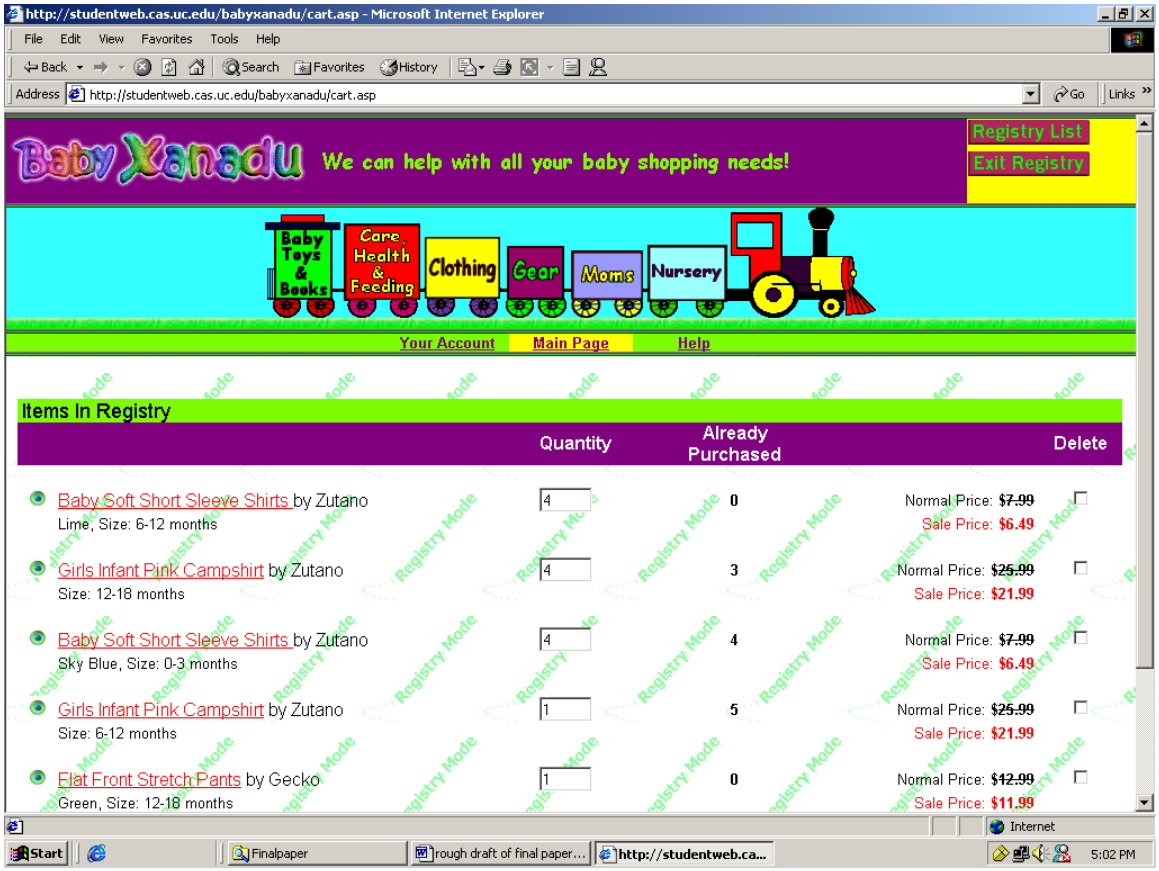


Figure 7. Cart.asp in registry mode.

5.5 Checkout

Checkout can be accessed from either the header or the *cart.asp* page. Both links are created by forms and have a buttons to be selected and actually point to *login.asp* with the mode variable being passed as "checkout". There are six steps in checking out. The first step is for the user to login (see Figure 8), if they aren't a member they must become one by hitting the New Member button, which takes them to the join.asp page (see Figure 9). Where by using a form and text boxes, the user enters name, address, email and password. They will then use their email address as their username and the password in login.asp to logon.

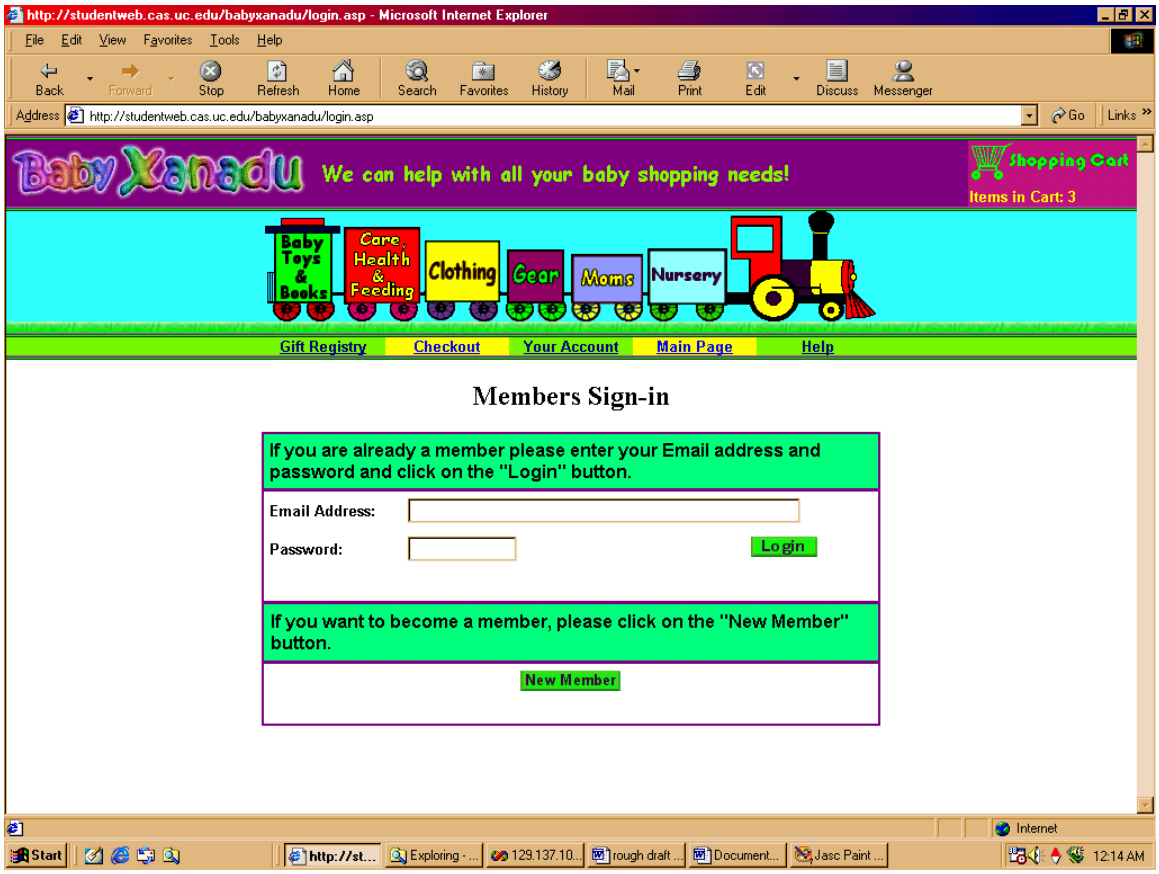


Figure 8. Users can login if they are a member or create a membership

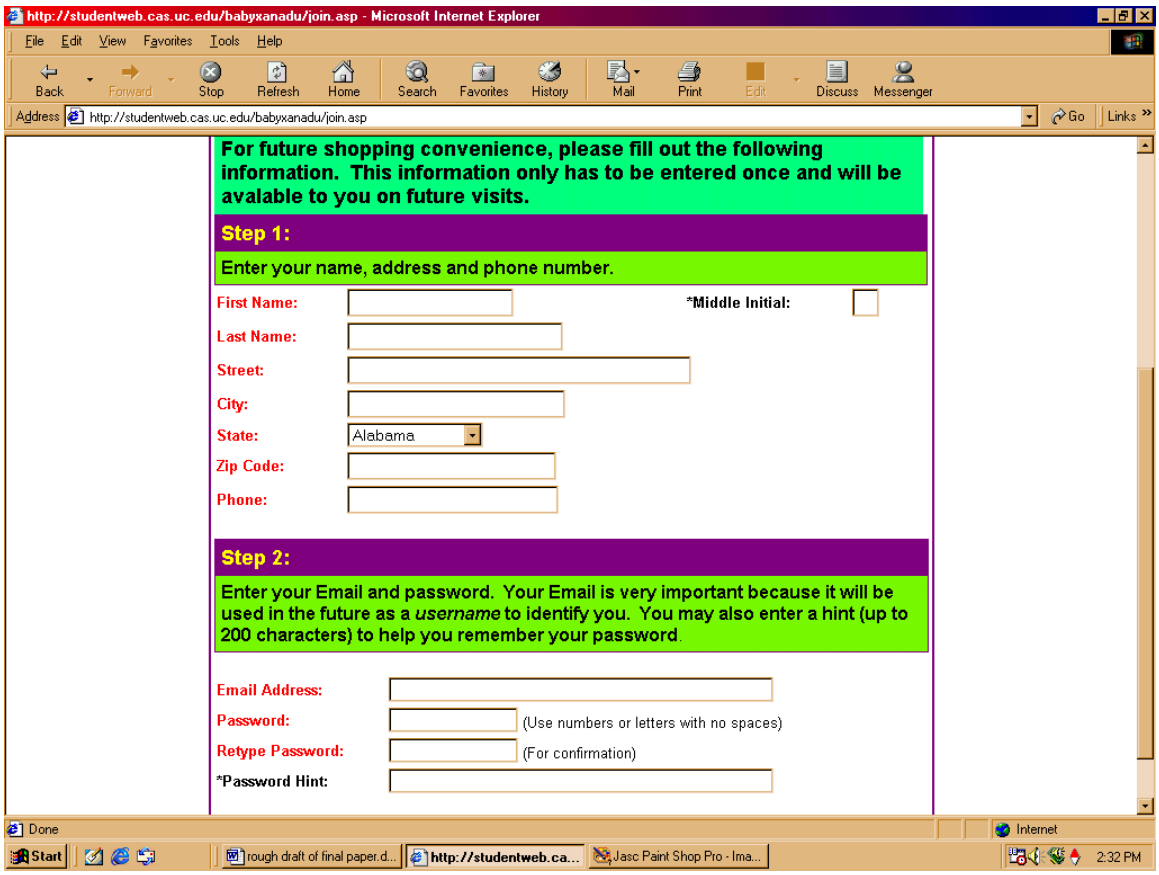


Figure 9. Users can become members by filling out the form on join.asp.

Every page in the checkout sequence uses forms to pass on information. Each page uses the ASP command Request.Form to assign the previous form's data into variables, which are then checked for validity. If any of the data is invalid, the page automatically redirects back to the previous page so the member can fix the information. The ASP command Response.Redirect is the method used for redirection. If the data is valid, then the variables are assigned to hidden text boxes within the form, so when the "Continue" button is selected, the data currently on the page and all the date from the previous pages are passed. By the end of the sequence, all the data previously collected has been passed in. This could also have been done by session variables, but session

variables tend to have a high memory overhead, especially if many users are also checking out.

The next page after the login is *chekout1.asp* (see Figure 10), where the member decides where they want to send the order. An ADO recordset is created from the *Customers* table, with the member's address as the data. This data is then assigned to variables and displayed as the billing address. Under the billing address are text boxes allowing the user to enter an alternate address. If the user has shopped from a registry, a session variable was created, that session variable is searched for, if it exists, then the user also will have the option of sending the order to the registry's address. The registry address is not actually shown, just the names of the parents and baby in the in the registry.

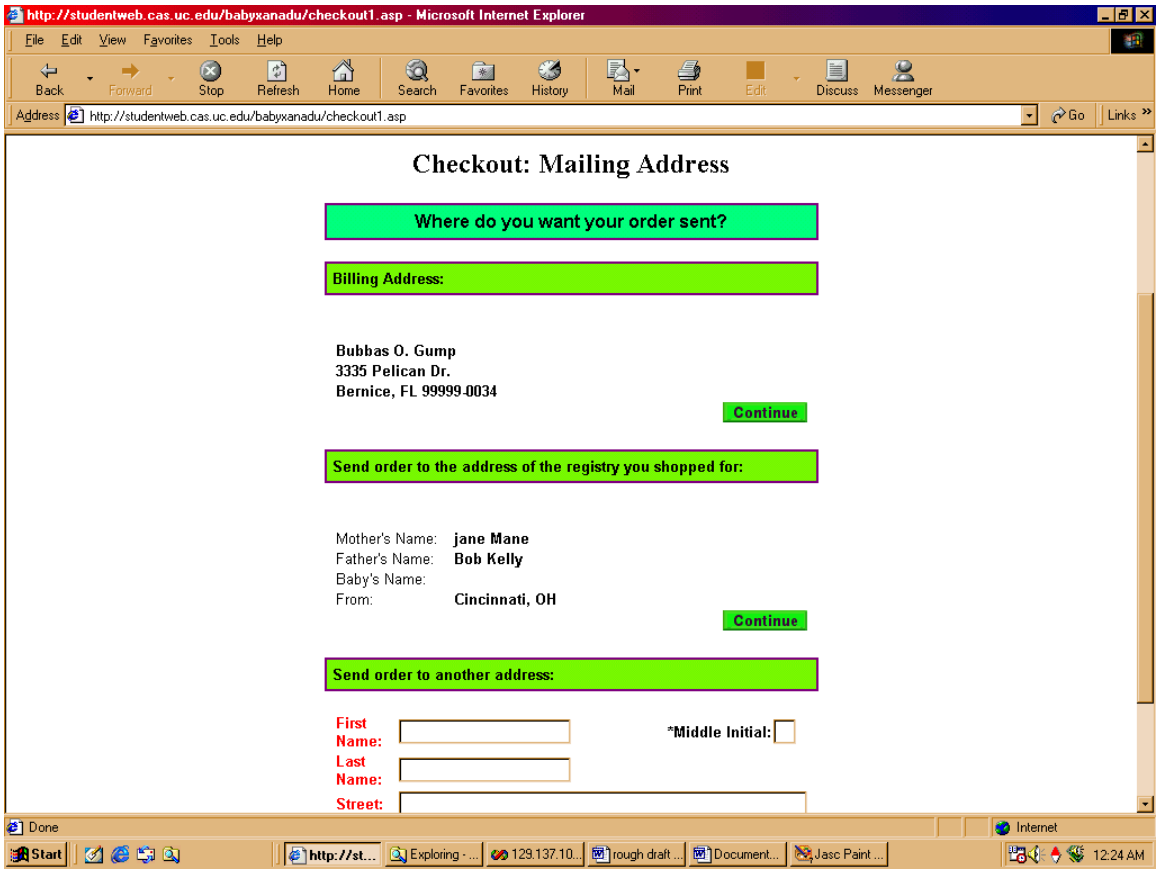


Figure 10. After logging in, the first step in checking out is choosing a mailing address.

The next page is *checkout2.asp* (see Figure 11), where credit card information is taken in. Any valid credit cards that the user has used will have their type and the last five digits displayed. This is done by using and ADO recordset and checking the database table, *CreditCards* for any credit cards attached to that member's customer ID. If the member doesn't want to or hasn't used a previous credit card, text boxes are made available to enter the information. This includes the credit card type (Visa, Master Card, American Express), its number and it expiration date.

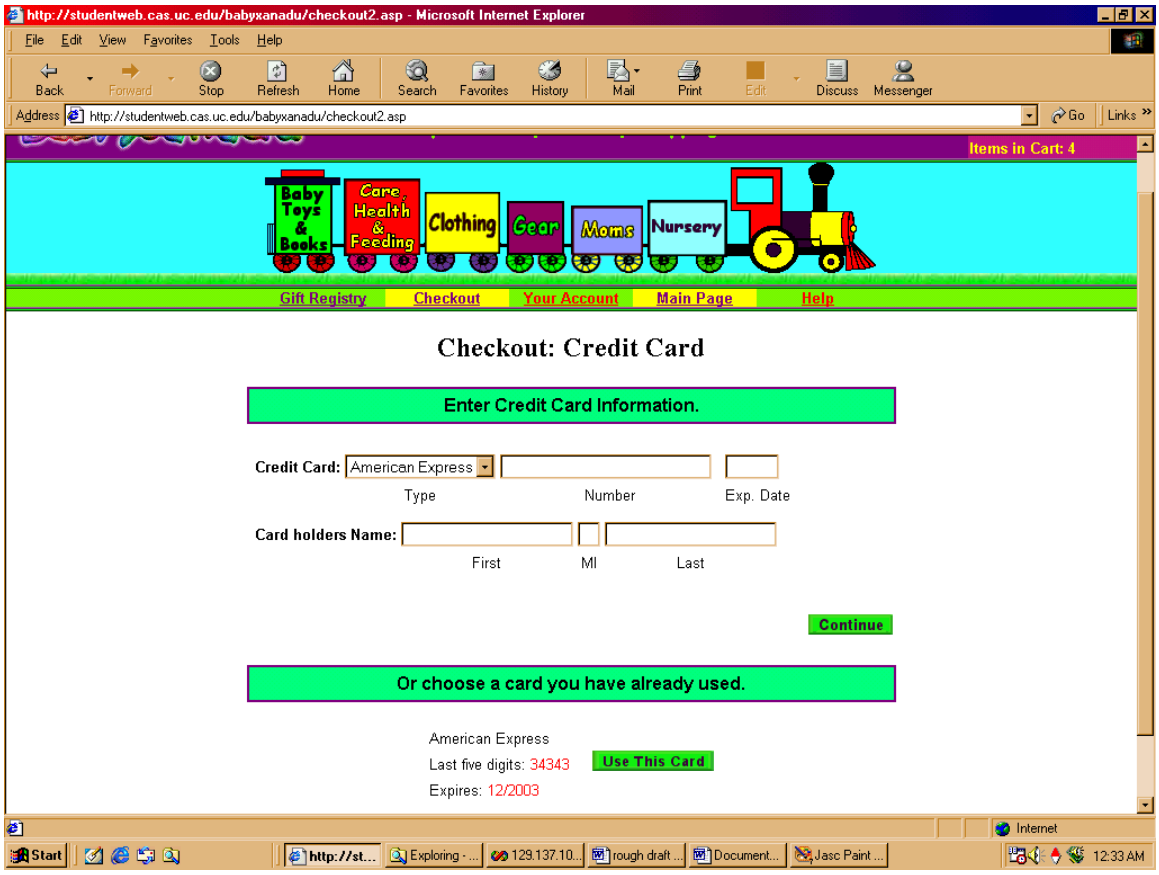


Figure 11. After choosing a mailing address, the credit card information is entered.

Once the credit card information is taken and "Continue" is selected, the next page *checkout3.asp* is loaded (see Figure 12). This is where the user sees what they have set for purchase, the total cost, and three shipping choices are given. The code to display

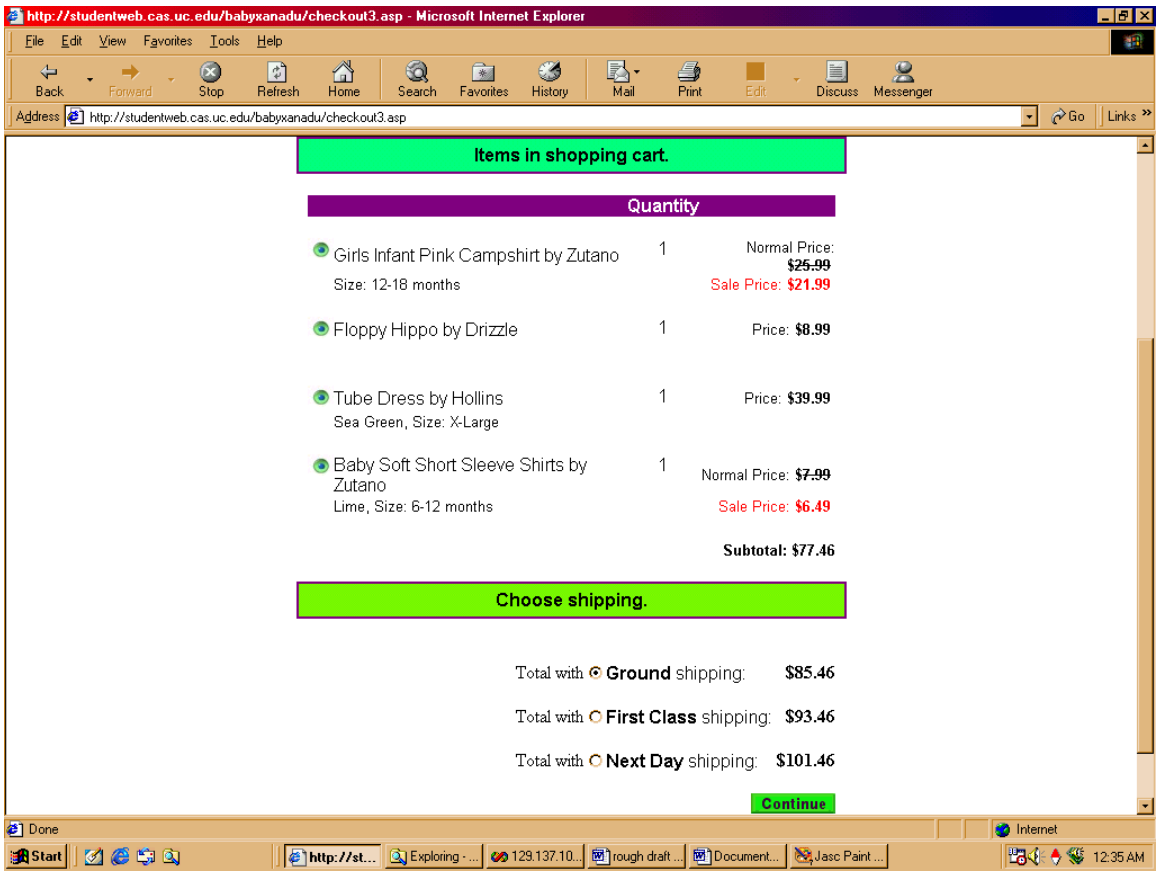


Figure 12. Shipping is next after credit cards in the checkout sequence.

the products is almost identical to the shopping cart, but the user doesn't have the option of deleting or changing quantities. Shipping choices are presented as radio buttons, and the total order cost is calculated and presented for each one. The user only needs to select one and press "Continue".

After the shipping is chosen, *checkout4.asp* is loaded (see Figure 13). This is only a confirmation page, it displays all of the information previously entered (all passed via forms) and shows all the products in the cart. If everything is satisfactory the user presses the "Process Order" button and *checkout5.asp* is called.

In *checkout5.asp* all of the information is put into ADO parameters and sent to the database, where the stored procedure ps_ProcessOrder inserts the data into the *Oders* and

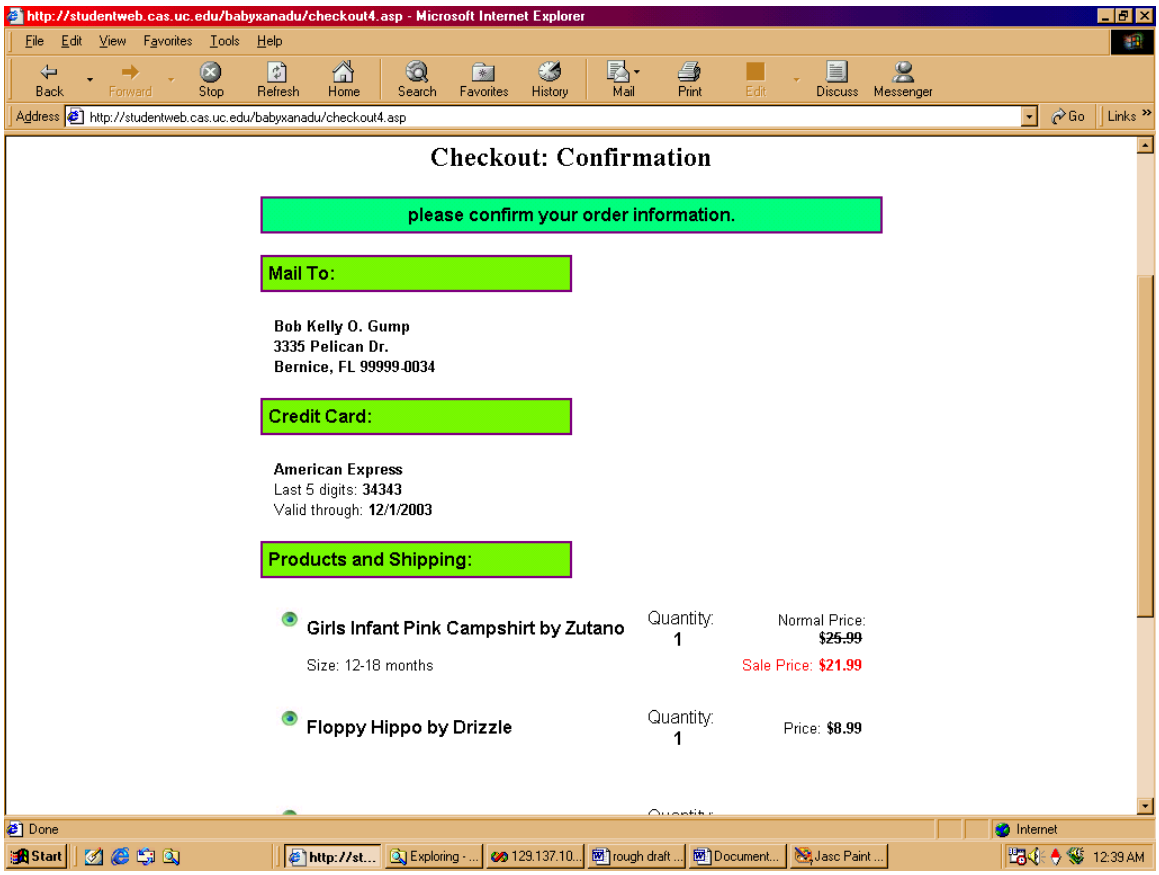


Figure 13. Order confirmation is the final step before confirming an order.

Orderdetails tables. Qualifying rows from the CartDetails or RegistryDetails are deleted, and the order ID is sent back and displayed on the screen. In addition an email is sent to the user verifying that an order has been processed and again giving the order ID. Finally all of the session variables are abandoned using the Session.Abandon ADO command.

6 Conclusions and Recommendations

The process of researching and developing this project has been very educational. I have greatly increased my knowledge of Microsoft's SQL Server 7.0 and my skills in databases in general. I had had some experience with ASP, VBScript and ADO, but I have learned quite a bit more. This project has given me a better idea on what is needed in an E-commerce site. I now understand the need for a good layout and easy and intuitive navigation. The addition of specials and good graphics are also often important. This is probably the biggest programming task I have undergone, and I have learned to allow extra time for debugging and just how to program more efficiently.

There are a few things that would need to be changed for an actual E-commerce site. This site does not have any security. A true E-commerce site would use Secure Socket Layer (SSL) to encrypt data, especially credit card information. Unfortunately a certificate must be purchased to use SSL and that can cost over $500, and that was not in the budget.

Another major factor also deals with credit cards. An E-commerce site must have some way of verifying and debiting credit cards. There are services that do this, but all of them require some type of merchant account and a gateway to a bank account. Some services will provide both, but all of them charge a fee that can range from a percentage of the total charged to a card, to a set amount for each transaction. All this costs money, which again is not in the budget, and is unrealistic for a strictly academic project.

Appendix A

Design and Development

1 Budget

1.1 Software

1.1.1 Web Development

Microsoft's Visual Interdev 6.0 was used to design and code the *Baby Xanadu* site pages. The estimated cost for this software is $469.00.

1.1.2 Graphic Development

To create the graphics for the site, two separate software titles were used. JASC's Paint Shop Pro 7.0 was used for the majority of graphics, buttons and graphic touch up, while ULEAD Cool 3D was used for the 3D "Baby Xanadu" title found in the header. The estimated cost for Paint Shop Pro is $84.99 and ULEAD Cool 3D is $39.95.

1.1.3 Database Development

Mircrosoft's SQL Server 2000 Developer was used to create the database. This is not the full version of SQL Server that at a minimum costs about $1200.00 for five clients. This version only cost 442.99.

1.2 Hardware

1.2.1 Development PC

A PC will be needed for development costing approximately $1000.00 for an Intel P3 800mhz.

1.2.2 Web and Database Server

Web space will be leased from an ISP along with a connection to a SQL server database server. The estimated cost is $149.00 a month for both.

1.3 Domain Registration

To register the domain name "BabyXanadu" for .com, .net and .org, the cost will be $210.00 for two years**.**

1.4 Budget Table see Figure 14.

Item	Monthly Cost	One Time Cost
Domain Registration		$210 for two years
Merchant Account Provider	2%-3% of Transactions	
Web and Database Hosting		$149.99
Microsoft Visual Interdev Software		469.00
Paint Shop Pro		84.99
ULEAD Cool 3D		39.95
Microsoft SQL Server 2000 Developer		442.99
PC		1000.00
Total		**$2185.93**

Figure 14. Budget Table

2 Timeline

From Senior Design 1 through Senior Design III each quarter various task had to be completed. Figures 15-17 show what was done each quarter related to time.

Item	Date to be Completed
Research E-commerce	10-27-00
Research needed software/hardware	11-5-00
Research budget	11-14-00
First draft of proposal due	11-16-00
Create proposal presentation	11-28-00
Proposal due	11-30-00
Present proposal to faculty	11-30-00

Figure 15. Senior Design I Time Table

Research database needs	1-10-01
Create basic database and populate with sample data	1-17-01
Meet with advisor for Progress Report 1	1-25-01
Design Web site layout	1-28-01
Create default page and product page	2-15-01
First draft of design freeze due	2-15-01
Create shopping cart page and members page	2-20-01
Create checkout pages	2-29-01
Final version of design freeze due	3-2-01
Create design freeze presentation	3-7-01
Present prototype to faculty	3-8-01

Figure 16. Senior Design II Time Table

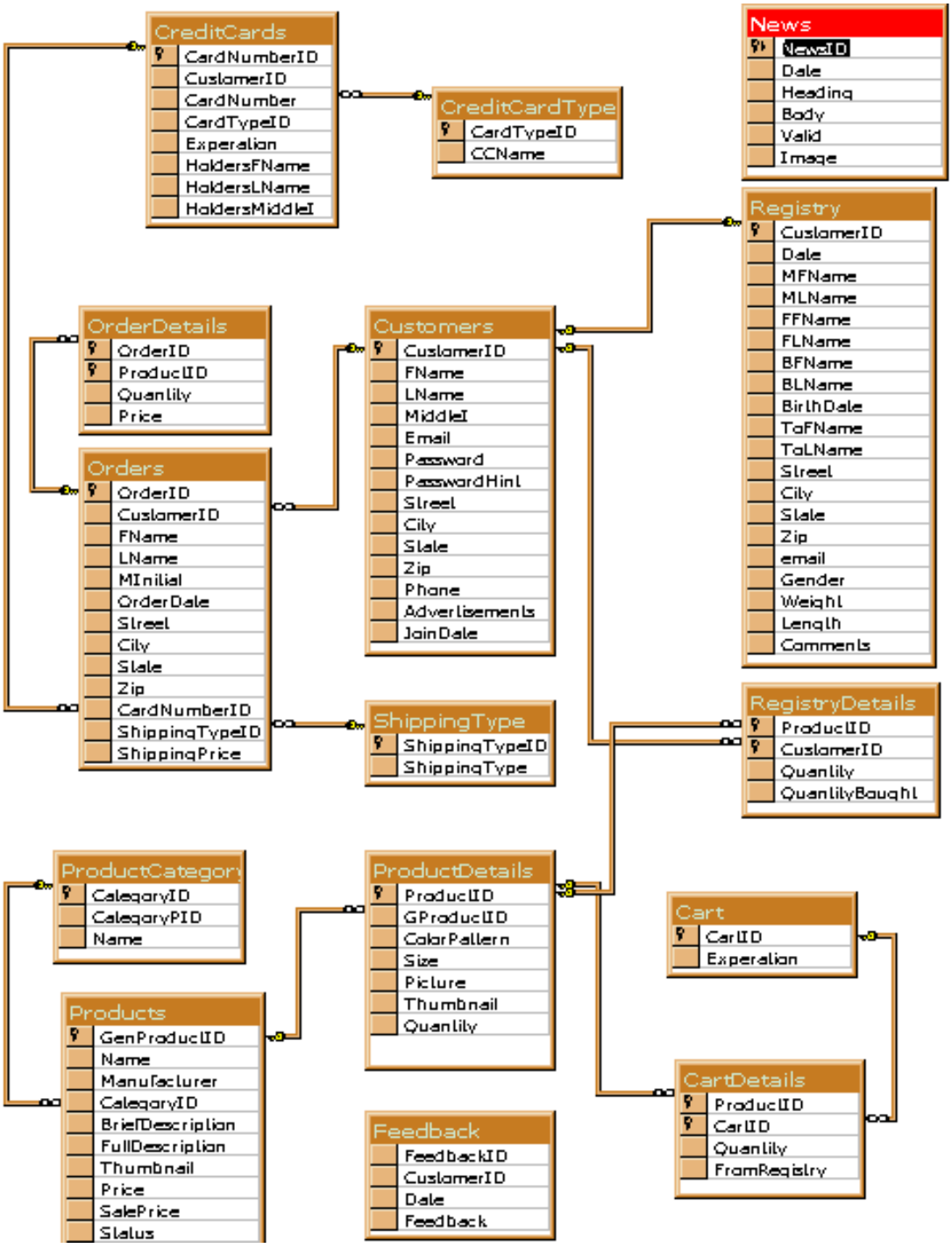
Create registry	3-24-01
Modify pages for registry	4-3-01
Create final graphics	4-14-01
Populate database	4-28-01
Tweak site	5-10-01
Tweak database	5-17-01
Create final presentation	5-22-01
Create final report	5-22-01
Present final report	6-01-01

Figure 17. Senior Design III Time Table

Appendix B

Database Information

1. Database Diagram



2. Stored Procedures

A variety of stored procedures were used during this project. Here are a few samples.

2.1 Process Order

When an order is finally ready to be processed, this procedure is used. Called in by checkout5.asp.

```
CREATE PROCEDURE ps_ProcessOrder
    (
            @CID int,
            @CCID int,
            @FName varchar(20),
            @MI char(1),
            @LName varchar(20),
            @Street varchar(50),
            @City varchar(25),
            @State char(2),
            @Zip varchar(12),
            @Shipping int,
            @ShippingPrice money,
            @CartID int,
            @RID int,
            @OrderID int output,
            @Email varchar(80) output
    )

AS

BEGIN TRAN

INSERT INTO Orders (CustomerID,FName,LName, MInitial, Street , City, State, Zip, CardNumberID,
ShippingTypeID, ShippingPrice)
VALUES
(@CID,@FName,@LName,@MI,@Street,@City,@State,@Zip,@CCID,@Shipping,@ShippingPrice)


SELECT @OrderID=@@Identity

SELECT CartDetails.CartID as OrderID, CartDetails.ProductID,CartDetails.Quantity, Products.SalePrice
AS Price INTO #TempTable
FROM Products INNER JOIN ProductDetails ON Products.GenProductID = ProductDetails.GProductID
INNER JOIN   CartDetails ON    ProductDetails.ProductID = CartDetails.ProductID WHERE
cartid=@CartID and saleprice>0

INSERT INTO #TempTable SELECT CartDetails.CartID as OrderID, CartDetails.ProductID,
CartDetails.Quantity, Products.Price
FROM Products INNER JOIN ProductDetails ON Products.GenProductID = ProductDetails.GProductID
INNER JOIN CartDetails ON ProductDetails.ProductID = CartDetails.ProductID WHERE
cartid=@CartID and ( not(saleprice>0) or saleprice is null)

UPDATE #TempTable SET OrderID=@OrderID

INSERT INTO orderdetails select * FROM #TempTable
```

```
DROP TABLE  #TempTable

IF @RID>0
BEGIN
     UPDATE registrydetails set QuantityBought=RegistryDetails.QuantityBought + CartDetails.Quantity
     FROM CartDetails INNER JOIN RegistryDetails ON CartDetails.ProductID =
     RegistryDetails.ProductID
     WHERE registrydetails.customerid=@CID  AND CartDetails.cartid=@CartID   AND
     cartDetails.fromregistry=1
END

DELETE FROM CartDetails WHERE CartID=@CartID

SELECT @Email=email FROM customers WHERE CustomerID=@CID

COMMIT TRAN
```

2.2 Customer Procedures

There are two stored procedures that process the *customers* table. The first procedure adds the information customer information and the second updates it.

2.2.1 Add Customer

```
CREATE PROCEDURE ps_AddCustomer

     (
               @FName varchar(20),
               @LName varchar(20),
               @MI char(1),
               @Email varchar(80),
               @Password varchar(12),
               @PwrdHint varchar(200),
               @Street varchar(50),
               @City varchar(25),
               @State char(2),
               @zip varchar(12),
               @Phone varchar(16),
               @Ads as bit

     )
AS

BEGIN TRAN

IF  ( Select count(CustomerID ) from customers where Email=@Email and Password=@Password) = 0
BEGIN
     INSERT INTO Customers (FName, LName, MiddleI, Email, Password, PasswordHint, Street, City,
     State, Zip, Phone, Advertisements)
     VALUES
     (@FName, @LName, @MI, @Email, @Password, @PwrdHint, @Street, @City, @State, @zip,
     @Phone, @Ads)
END

COMMIT TRAN
```

2.2.2 Update Customer

```
CREATE PROCEDURE ps_UpdateCustomer

    (
            @CID int,
            @FName varchar(20),
            @LName varchar(20),
            @MI char(1),
            @Email varchar(80),
            @Password varchar(12),
            @PwrdHint varchar(200),
            @Street varchar(50),
            @City varchar(25),
            @State char(2),
            @zip varchar(12),
            @Phone varchar(16),
            @Ads as bit
    )
AS

BEGIN TRAN

UPDATE  Customers set FName=@FName, LName=@LName, MiddleI=@MI, Email=@Email,
Password=@Password, PasswordHint=@PwrdHint, Street=@Street, City=@City, State=@State,
Zip=@zip, Phone=@Phone, Advertisements=@Ads
 WHERE customerid=@CID

COMMIT TRAN
```

Appendix C

Code

1. Default Page

To get an understanding of the programming involved for this project, here is the code used to create the default page, *default.asp.*

```
<%@ Language=VBScript %>
<!-- #include file=ADOVBS.INC -->
<!-- #include file=inc_dbconnection.asp -->
<!-- #include file=inc_CatArray.asp -->
<!-- #include file=inc_CategoryPath.asp -->

<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<%
'************************************************************************
'** This procedure will display products of that category ***
sub subDisplayProducts (intIndex)
dim intCategoryID 'will hold the current category id
dim pg 'page of products
dim strTNail

    intCategoryID=application("productcategories")(0,intIndex)

    pg = TRIM(request("pg"))
    if pg="" then pg=1

    'open the recordset
    objRst.activeconnection=ConSTring
    objRst.CursorType=adOpenStatic
    objRst.CursorLocation=adUseClient
    objRst.PageSize=10 ' change this later

    strSQL= "Select * from vw_ShowGenProd where CatID=" & intCategoryID

    objRst.open strSQL



    if not objRST.eof then objRst.AbsolutePage=pg

%>
    <table border="0" cellpadding="3" cellspacing="0">

<%  'put links to next or previous page if any
    if objRst.Pagecount > 1 then
%>          <tr><td colspan="3">
            <table align="right">
                <tr>
<%                  if Pg > 1 Then
                        strPrev="<B><font face=""arial,helvetica,sans-serif"" size=""1""><A
href=""default.asp?CatIndex=" & intIndex & "&pg=" & cInt(pg)-1 & """> < Previous </A></font></B> "
%>                      <td align="right"><%=strPrev%></td>
<%                  end if
                    if (objRst.pagecount) > cInt(pg) Then
                        strNext="<B><font face=""arial,helvetica,sans-serif"" size=""1""><A
href=""default.asp?CatIndex=" & intIndex & "&pg=" & cInt(pg)+1 & """> Next ></A></Font><B> "
%>                      <td align="Left"><%=strNext%></td>
<%                  end if
%>              </tr>
```

```
                </table>
        <%  end if

                'show product picture, name, manufacture, prices and description
                do While (NOT objRst.eof) and (intProdCount < objRst.pagesize)
                        intProdCount=intProdcount+1
                        if objRst("Status")< 3 then 'status 3 means not to show product
        %>                      <tr>
                                        <td> <!-- product thumbnail -->
        <%                                      if objRst("TNail")="??" or isnull(objRst("TNail")) or objRst("TNail")="" then
                                                        strTNail="NoPict.gif"
                                                else
                                                        strTNail=objRst("TNail")
                                                end if
        %>                                      <a href="product.asp?GPID=<%=objRst("GPID")%>&amp;CatIndex=<%=intIndex%>&amp;pg=<%=pg%>"><img SRC="i/T/<%=strTNail%>" border="0"></a>
                                        </td>

                                        <td width="100%" valign="top" align="left">

                                                <table width="100%" valign="top" cel cellspacing="0" cellpadding="0">
                                                        <tr bgcolor="LawnGreen">
                                                                <td align="left" valign="top"><a href="product.asp?GPID=<%=objRst("GPID")%>&amp;CatIndex=<%=intIndex%>&amp;pg=<%=pg%>"><font face="arial,helvetica,sans-serif" size="2"><b><%=objRst("Name")%></b></a>
                                                                        by <%=objRst("Manufacturer")%></font></td>
                                                                <td valign="top" align="right" width="70"><font face="arial,helvetica,sans-serif" size="2"><b>$<%=objRst("Price")%></b></font></td>
                                                        </tr>
                                                        <tr><td valign="top" align="left"><font face="arial,helvetica,sans-serif" size="2"><%=objRst("BDes")%></font></td>
        <%                                                      if objRst("SPrice")<>"" then

        %>                                                              <td valign="top" align="right" width="70"><font face="arial,helvetica,sans-serif" size="2" color="red">Sale <b>$<%=objRst("SPrice")%></b></font></td>
        <%                                                      end if
        %>                                              </tr>
                                                </table>

                                        </td>
                                </tr>
        <%              end if
                        objRst.movenext
                loop
        %>  </table>
        <%
                'display an index to possible pages
                if objRst.Pagecount>1 then
        %>              <center><p>
                        <font face="arial,helvetica,sans-serif" size="2">
                        <b>Go to page: </b>
        <%
                        for i=1 to objRst.Pagecount
                                if i <> cINT(pg) THEN
                                        strPages=strPages & "<A href=""default.asp?CatIndex=" & intIndex & "&pg=" & i & """>" & i & "</A> "
        %>                              <a href="default.asp?CatIndex=<%=intIndex%>&amp;pg=<%=i%>"><%=i%></a> 
        <%                      else
                                        strPages=strPages & "<B>" & i & "</B> "
        %>                              <b><%=i%></b> 
        <%                      end if
                        next
        %>              </font>
                        </p>
                        </center>
```

```
<%  end if
    objRst.close
end sub
%>
<%
'************************************************************************
' This function will  tell if a category has children
' iIndex is the current index of the category array
' intArraySize is the size of the category array
function funChildren(iIndex,intArraySize)
dim intCurID

    funChildren=false

    if iIndex=-1 then funChildren=true 'main page, main categories

    i=iIndex + 1

    if iIndex=-1 then
        intCurID=application("productcategories")(0,0)
    else
        intCurID=application("productcategories")(0,iIndex)
    end if

    do while (i <= intArraySize)
        if application("productcategories")(1,i)=intCurID then
            funChildren=True
            exit do
        end if
        i=i+1
        if (i<=intArraySize) then
        'exit out of loop if no possibility of finding child
            if (intCurID < application("productcategories")(1,i)) then
                exit do
            end if
        end if
    loop
end function
%>

<%
'************************************************************************
'this procedure uses recursion to find 2 levels of catalog tree
sub subListChildren (intIndex,intArraySize,intLevel)
    dim bolFound 'is set to true if a Parent finds a child
        bolFound=false

        for j=intIndex+1 to intArraySize
            if application("productcategories")(1,j)=application("productcategories")(0,intIndex) then
                bolFound=true

                if intlevel=1 then 'show parent
%>
                    <tr>
                        <td valign="top" width="10"><img SRC="i/dot.gif" BORDER="0"></td>
                        <td width="100%" colspan="2" align="left"><font face="arial,helvetica,sans-serif" size="2"><b><a href="default.asp?CatIndex=<%=j%>">
                        <%=application("productcategories")(2,j)%></a></b></font>
<%                      subListChildren j,intArraySize,2
%>                      </td>
                    </tr>
<%
                else 'else show children
%>                  <tr>
                        <td> </td>
                        <td width="5" valign="top"><font color="purple"><b> -</b></purple></td>
                        <td align="left" width="100%"><font face="arial,helvetica,sans-serif" size="2"><a href="default.asp?CatIndex=<%=j%>">
```

```
        <%=application("productcategories")(2,j)%></a></font>
                                                        </td>
                                                </tr>
    <%                          end if
                        elseif bolFound=true then
                                exit for
                        end if
                next
    end sub
    %>
    <%
    '*****************************************************************************
    'this procedure creates an array of all the categories without children below
    'the specified main category, uses recursion to accomplish this
    sub subFindAllChildren(intCIndex,intCAmount,arrArray,intNewIndex)
        dim intCurCategory
        dim bolIsChild


        intCurCategory=application("productcategories")(0,intCIndex)

        for i=intCIndex + 1 to intCAmount step 1
                if application("productCategories")(1,i)=intCurCategory then
                        bolChild=true
                        subFindAllChildren i,intCAmount,arrArray,intNewIndex 'recursive call
                end if
        next

        '***
        if (not bolIsChild) then
                if intNewIndex=0 then
                        redim arrArray (0)
                        arrArray(0)=intCurCategory
                else
                        redim preserve arrArray(intNewIndex)
                        arrArray(intNewIndex)=intCurCategory
                end if
                intNewIndex=intNewIndex+1
        end if



    end sub
    %>
    <%
    '*****************************************************************************
    'This procedure print the specials products that are passed to it in an array.  ****
    '*****************************************************************************

    sub subDisplaySpecials(arrSpecialArray,intCInx,j)

    %>
        <table>
                <tr>
                        <td> <!-- Product Picture -->
                                <a
href="product.asp?GPID=<%=arrSpecialArray(0,j)%>&amp;CatIndex=<%=intCInx%>">
                                <img src="i/T/<%=arrSpecialArray(4,j)%>" border="0"></a>
                        </td>
                        <td valign="bottom"><!-- Name and Manufacturer -->
                                <b><font face="arial,helvetica,sans-serif" size="3">
                                <a
href="product.asp?GPID=<%=arrSpecialArray(0,j)%>&amp;CatIndex=<%=intCInx%>">
                                <%=arrSpecialArray(2,j)%> by <%=arrSpecialArray(3,j)%></a></font></b>
                        </td>
                </tr>
                <tr>
                        <td colspan="2"> <!-- Brief Description -->
                                <b><font face="arial,helvetica,sans-serif" size="3">
```

```
                                        <%=arrSpecialArray(5,j)%></font></b>
                                </td>
                        </tr>
                        <tr>
                                <td align="right" colspan="2"><b><font face="arial,helvetica,sans-serif" size="3">
                                        <strike>$<%=arrSpecialArray(6,j)%></strike><font
color="red">  $<%=arrSpecialArray(7,j)%>
                                        </font></font></b>
                                </td>
                        </tr>
        </table>

        <%
        end sub
        %>

        <%
        '*************************************************************************************
        'this procedure will create the specials application array
        'products that are on sale will be put in this array

        sub FindSpecials(intCIndex,intCatAmount)

        dim strSQL
        dim intSpecAmount
        dim arrSpecials 'local specials array
        dim arrArray
        dim arrFinalArray
        const intNumberShown=4


                Randomize
        application("specials")=""

                IF NOT isArray(Application("Specials")) then
                        'build query with 8 columns
                        strSQL                                                  =                                       "Select
GenProductID,CategoryID,Name,Manufacturer,Thumbnail,BriefDescription,Price,SalePrice " &_
                        "FROM Products WHERE Status=1 ORDER BY CategoryID,Name"

                        objRSt.OPEN strSQL,constring,3,3

                        IF not objRST.EOF THEN
                                arrSpecials=objRST.getrows
                                objRST.close
                        end if

                        'create and add to specials application array
                        Application.Lock
                        Application("Specials")=arrSpecials
                        Application.UnLock
                ELSE
                        arrSpecials=Application("Specials")
                END IF

                intSpecAmount=ubound(arrSpecials,2)
                'find all subcategories of current category

                if intCIndex > -1 then 'not first page
                        subFindAllChildren intCIndex,intCatAmount,arrArray,0
                        if isArray(arrArray) then
                                intNewIndex=0
                                for i=0 to ubound(arrArray)

                                        for j=0 to ubound(arrSpecials,2)
                                                if arrArray(i)=arrSpecials(1,j) then
                                                        if intNewIndex=0 then
                                                                redim arrFinalArray(7,0)
                                                                arrFinalArray(0,0)=arrSpecials(0,j)
                                                                arrFinalArray(1,0)=arrSpecials(1,j)
```

```
                                                arrFinalArray(2,0)=arrSpecials(2,j)
                                                arrFinalArray(3,0)=arrSpecials(3,j)
                                                arrFinalArray(4,0)=arrSpecials(4,j)
                                                arrFinalArray(5,0)=arrSpecials(5,j)
                                                arrFinalArray(6,0)=arrSpecials(6,j)
                                                arrFinalArray(7,0)=arrSpecials(7,j)
                                        else
                                                redim preserve arrFinalArray(7,intNewIndex)
                                                arrFinalArray(0,intNewIndex)=arrSpecials(0,j)
                                                arrFinalArray(1,intNewIndex)=arrSpecials(1,j)
                                                arrFinalArray(2,intNewIndex)=arrSpecials(2,j)
                                                arrFinalArray(3,intNewIndex)=arrSpecials(3,j)
                                                arrFinalArray(4,intNewIndex)=arrSpecials(4,j)
                                                arrFinalArray(5,intNewIndex)=arrSpecials(5,j)
                                                arrFinalArray(6,intNewIndex)=arrSpecials(6,j)
                                                arrFinalArray(7,intNewIndex)=arrSpecials(7,j)
                                        end if
                                        intNewIndex=intNewIndex+1
                                end if
                        next

                next

                if IsArray(arrFinalArray) then
                '*** show specials but not in main page ***

                        '*** randomly choose specials to show ***
                        intArrSize=ubound(arrFinalArray,2)+1
                        skip=intArrSize/intNumberShown
                        if intArrSize<=intNumberShown then skip=1
                        bolNewRow=TRUE

%>                      <table border="0" cellspacing="10" colspacing="7" width="95%">
                                <tr>
                                        <td>
                                                <font                   face="arial,helvetica,sans-serif"
color="YellowGreen" size="4">
                                                <b><i>Heres a few of our specials!</font></i></b>
                                        </td>
                                </tr>
<%
                        for i=0 to intArrSize -1 step skip
                                offset=RND * (skip-1)
                                j=i+offset

                                if bolNewRow then
                                        Response.Write "<TR><TD align=Left>"
                                        subDisplaySpecials arrFinalArray,intCIndex,j
                                        Response.Write "</TD></TR>"
                                        bolNewRow=FALSE
                                else
                                        Response.Write "<TR><TD align=Right>"
                                        subDisplaySpecials arrFinalArray,intCIndex,j
                                        Response.Write "</TD></TR>"
                                        bolNewRow=TRUE
                                end if

                        next
                        Response.Write "</TABLE>"
                end if
        end if

    else 'main page so use other array...
        intArrSize=ubound(arrSpecials,2)

        '*** randomly choose specials to show ***
        intArrSize=ubound(arrSpecials,2)+1
        skip=intArrSize/intNumberShown
        if intArrSize<=intNumberShown then skip=1
        bolNewRow=TRUE
```

```
%>              <table border="0" cellspacing="10" colspacing="7" width="95%">
                        <tr>
                                <td>
                                        <font face="arial,helvetica,sans-serif" color="YellowGreen" size="4">
                                        <b><i>Heres a few of our specials!</font></i></b>
                                </td>
                        </tr>
<%
                for i=0 to intArrSize -1 step skip
                        offset=RND * (skip-1)
                        j=i+offset
                        if bolNewRow then
                                Response.Write "<TR><TD align=left>"
                                subDisplaySpecials arrSpecials,intCIndex,j
                                Response.Write "</TD></TR>"
                                bolNewRow=FALSE
                        else
                                Response.Write "<TR><TD align=right>"
                                subDisplaySpecials arrSpecials,intCIndex,j
                                Response.Write "</TD></TR>"
                                bolNewRow=TRUE
                        end if

                next
                Response.Write "</TABLE>"

        end if



END SUB
%>
<%
'*****************************************************************************
'*** This procedure will pull the news from the database and display it.       ****
'***                                                                          ****
sub subDisplayNews

dim arrNews
dim intTotal
application("arrnews")=""
        if not isarray (application("arrNews")) then
                '*** put in application array if there isn't one ***
                strSQL="select Heading,Body,Image from news where Valid=0 order by date desc"
                objRST.open strSQL,constring,adOpenForwardOnly,adLockReadOnly
                if not objRST.eof then
                        application.lock
                        application("arrNews")=objRST.getrows
                        application.UnLock
                end if
        end if
        arrNews=application("arrNews")

        intTotal=ubound(arrNews,2)

%>  <table bgcolor="Aqua">
<%  for i=0 to intTotal step 1
%>              <tr>
                        <td><font face="arial,helvetica,sans-serif" color="purple" size="2">
                                <b><%=arrNews(0,i)%>:</b> <%=arrNews(1,i)%></font>
                        </td>
                </tr>
                <tr><td> </td></tr>
<%  next
%>  </table>
<%

end sub
%>
```

```
<title>Welcome to Baby Xanadu!</title>
</head>

<!-- ************************** begin body ********************************************** -->
<!-- ************************** begin body ********************************************** -->



<%
dim intCatIndex         'will hold current index
dim intCatAmount        'will hold the max index for category array
dim bolCategories       'if this ends up holding a false value, then products should be displayed

intCatIndex=trim(request("CatIndex"))
if isnull(intCatIndex) or intCatIndex="" or intCatIndex="-1" then intCatIndex=-1

intCatAmount=ubound(application("ProductCategories"),2)
bolCategories=funChildren(intCatIndex,intCatAmount)
strHelpAnchor="Navigation"

%>

<!-- #include file=inc_header.asp -->


<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<%
    if intCatIndex=-1 then
    ' special circumstances, these are main categories, on main page

%>          <td width="156" bgcolor="PaleGreen" valign="top">
                <table width="156" border="0" cellspacing="0" cellpadding="0">
<%
                        i=0
                        do while application("ProductCategories")(1,i)=0 and not(i>intCatAmount)

%>                          <tr>
                                <td valign="top" width="10"><img SRC="i/dot.gif"
BORDER="0" WIDTH="10" HEIGHT="14"></td>
                                <td colspan="2">
                                    <font face="arial,helvetica,sans-serif"
size="2"><b>
                                    <a
href="default.asp?CatIndex=<%=i%>"><%=application("productcategories")(2,i)%></a></b></font>
                                </td>
                            </tr>
<%                          subListChildren i,intCatAmount,2
%>

<%                          i=i+1
                        loop
%>
                        <tr><td colspan="4"><p> </p></td></tr>
                </table>
            </td>
<%  elseif bolCategories then
    'has children but isnt main category
%>
            <td width="156" bgcolor="PaleGreen" valign="top">
                <table width="156" border="0" cellpadding="0" cellspacing="0">
                    <tr>
                        <td colspan="3" valign="middle" align="left">
                            <table cellpadding="3" width="100%"><tr><td>
                            <font face="arial,helvetica,sans-serif"
size="3"><b><%=application("productcategories")(2,intCatIndex)%></b></font>
                            <hr>
                            </td></tr>
                            </table>
```

```
                                        </td>
                                    </tr>
    <%                              subListChildren intCatIndex,intCatAmount,1
    %>
                                    <tr><td colspan="4"><p> </p></td></tr>
                                </table>
                            </td>
    <%  end if

        if bolCategories then 'these 3 columns are borders
    %>          <td bgcolor="Purple" width="2"><img SRC="i/2xclear.gif" border="0" WIDTH="2" HEIGHT="1"></td>
                <td bgcolor="Lawngreen" width="2"><img SRC="i/2xclear.gif" border="0" WIDTH="2"
HEIGHT="1"></td>
                <td bgcolor="Purple" width="2"><img SRC="i/2xclear.gif" border="0" WIDTH="2" HEIGHT="1"></td>
    <%  end if
    %>

        <td valign="top" align="left" width="100%">
            <table border="0" width="100%" cellpadding="0" cellspacing="2">
    <%          if not intCatIndex=-1 then 'print path if not main page
    %>              <tr>
                        <td>
    <%                  subCreatPath intCatIndex,false
    %>                  </td>
                    </tr>
                    <tr>
                        <td bgcolor="Purple" colspan="2"><img SRC="i/2xclear.gif"
height="2"></td>
                    </tr>
    <%          else
    %>              <tr>
                        <td> 
                        </td>
                    </tr>
    <%          end if

                if not bolCategories then 'there are no subcategories, so display products
    %>              <tr>
                        <td>
    <%                  subDisplayProducts intCatIndex
    %>                  </td>
                    </tr>
    <%              else  'there are subcategories, so show specials
                        if intCatIndex=-1 then
                            '*** in main page, print welcom
    %>                      <tr>
                                <td colspan="2"><p><font face="arial,helvetica,sans-serif"
color="purple">
                                  <font size="9">W</font><font size="8">elcome
to Baby Xanadu!</font></p>
                                </td>
                            </tr>
                            <tr>
                                <td>
    <%                              FindSpecials intCatIndex,intCatAmount
    %>                              </td>
                                <td width="200" valign="top">
                                    <table width="100%" cellspacing="2"
bgcolor="purple">
                                    <tr>
                                        <td>
    <%                                      subDisplayNews
    %>                                  </td </TR>
                                    </table>

                                </td>
                            </tr>

    <%                  else
    %>                      <tr>
```

```
                                                <td colspan="2"><p><font face="arial,helvetica,sans-serif"
color="purple">
                                                  <font
size="8"><%=application("productcategories")(2,intCatindex)%></font></font></p>

                                                </td>
                                        </tr>

                                        <tr>
                                                <td>
        <%                                              FindSpecials intCatIndex,intCatAmount
        %>                                      </td>

                                                </td>

                                                <td width="200" valign="top">
                                                        <table          width="100%"            cellspacing="2"
bgcolor="purple">
                                                        <tr>
                                                                <td>
        <%                                                              subDisplayNews
        %>                                                      </td </TR>
                                                        </table>

                                                </td>

                                        </tr>
        <%
                                end if
                        end if
        %>
                </table>
            </td>

        </tr>
        <tr>
            <td colspan="4" bgcolor="Purple"><img SRC="i/2xclear.gif" height="2"></td>
        </tr>

        </table>
        </body>
</html>
```

References

1. 15 Seconds. http://www.15seconds.com

2. Amundsen, Michael. *Practical Visual InterDev 6.* New York: Que, 1999

3. DevGuru. http://www.devguru.com

4. E-CommerceTimes. http://www.ecommercetimes.com.

5. Levine, Jonathan and Stephen Walther. *Sams Teach Yourself E-Commerce Programming with ASP in 21 Days.* New York: Sams, 2000

6. Mahoney, Michael. "U.S. Reports Q3 E-Commerce Surge". http://www.ecommercetimes.com/ratescompare.asp. November 28, 2000

7. Reynolds, Mathew *Beginning E-Commerce with Visual Basic, ASP, SQL Server 7.0 and MTS* . New York: Wrox Press Inc, 2000

8. Wynkoop, Stephen. *Using Microsoft SQL Server 7.0.* New York: Que, 1999