Developing an E-Commerce System Using Active Server Pages

Robert W. Ingram
Ross-Culverhouse Chair
Culverhouse School of Accountancy
University of Alabama
Box 870220
Tuscaloosa, AL 35487
(205) 348-2907
ringram@cba.ua.edu

Dale L. Lunsford
Assistant Professor
Culverhouse School of Accountancy
University of Alabama
Box 870220
Tuscaloosa, AL 35487
(205) 348-5780
dlunsfor@cba.ua.edu

August 2002

Developing an E-Commerce System Using Active Server Pages

Abstract: This paper describes a case to illustrate analysis, design, and implementation issues involving a multi-tier e-commerce system. The case is designed for use in advanced accounting systems or systems analysis and design courses. The case involves analysis of a sales order system that will be implemented using a web interface and relational database, conceptual design of the system, and implementation of the system. A variety of tasks are involved in the case, but an instructor can select the tasks of relevance in a particular course.

Key Words: E-commerce, systems analysis and design, systems development, security.
Data Availability: All data and programs described are available from the authors.

Developing an E-Commerce System Using Active Server Pages

I. Introduction

Web-based applications are an important means of accessing business information. E-commerce represents an increasingly important part of the economy and is an obvious source of demand for web-based systems. In addition, companies may use the Internet for internal communications because of the ability of employees to access data from locations throughout the world. Companies like Oracle and SAP are increasingly focusing on the Web as a means of deploying their software products.

This paper describes a case that can be used in an advanced accounting information systems or systems analysis and design course to illustrate analysis, design, and implementation issues involving a multi-tier e-commerce information system. The case involves three sets of tasks: analysis of a sales order system to be implemented using a web interface and a database backend, conceptual design of the system, and physical design and implementation of the system. An instructor may choose to use all or any subset of the tasks.

The case provides for a variety of learning experiences. In the analysis stage, the case requires an examination of a company's needs to develop a web-based sales ordering system. Analysis skills students should gain from this case include:

- analysis of business processes,
- analysis of data and information requirements,
- identification of controls required in a web-database application, and
- using UML (Unified Modeling Language) to model requirements.

In the conceptual design stage, the case requires students to develop an implementation plan that describes the components of the system and how they interact. Design skills students should gain from this case include:

- data normalization,
- design of a multi-tier information system, and
- design of controls and security measures in a web-database application.

In the physical design and implementation stage, the case requires development of a user interface, a database backend, and a business logic component to link the user interface and the database. Physical design and implementation skills students should gain from this case include:

- development of relational database systems,
- development of web-based systems,
- programming business logic,
- linking web and database systems using ASP (Active Server Pages) middleware,
- implementation of application controls to ensure data integrity, and
- use of SQL (Structured Query Language) to interact with databases.

Depending on how the instructor chooses to use the case, students may develop skills in the following areas:

- use of CASE (Computer Aided Systems Engineering) tools,
- presentation of technical information, and
- preparation of written reports and documentation.

Which tasks are emphasized depends on the instructor's goals for the case and the course in which it is used. Basic familiarity with relational databases, HTML (Hypertext Markup Language), and SQL are assumed.

Teaching notes describe the use of the Al's Sounds case in an advanced AIS (Accounting Information Systems) course. Appendix A provides web addresses for the online demonstration of the system and source files, as well as resources and technologies discussed in the paper. Appendix B lists articles students can read to develop background

knowledge useful for completing this case. Appendix C presents an overview of UML. Appendix D discusses security issues in an e-commerce setting. Appendix E describes a sample implementation for the Al's Sounds case.

II. Case Description

Al's Sounds (AS) is a small producer of music CDs, marketed under the AS label. The company operates a recording studio and produces, markets, and distributes CDs for a small list of local artists. AS customers are regional distributors who handle sales to retailers in various parts of the country. AS customers place orders by phone to a sales clerk who completes a sales order form and forwards a copy to shipping. Shipping assembles orders and transfers them to commercial carriers for shipment. A copy of the shipping notice is forwarded to the accounts receivable clerk who compares the shipping notice with a copy of the sales order and who records the receivable and prepares an invoice.

AS is owned and managed by Al Trumpet. Distributors have requested that Al automate his sales order process to permit distributors to place orders using the Internet. Towards this end, he has contracted with your employer, Delbert Company, an accounting and consulting firm, to create a business to business (b2b) E-commerce system. As an initial step, he has asked for a web-based sales order system that permits approved distributors (AS customers) to place orders from a web browser. You have been assigned the task of preparing an analysis of AS system requirements and developing a prototype system to meet these requirements. Your tasks include:

1. Preparing an analysis of the system that describes company and customer needs and how the system will meet these needs.
2. Creating a conceptual design for the system that explains the operation of the system, specific activities the system will perform, and issues that will need to be considered in implementing the system.
3. Developing an ordering system that implements your design. The implementation will involve a three-tier architecture with a web-based interface and a relational

database. You will implement the prototype using ASP middleware and Access as a database management system.

Your review of the company has produced the following information. AS currently sells to a limited number of customers. Sample data about customers are provided in Table 1. These customers, plus any new distributors AS acquires, will need to log onto a web interface and place orders. The orders must indicate the titles and quantities of CDs ordered. Table 2 provides a list of several of the products AS currently sells. The prices are fixed and AS pays the shipping costs. Goods are shipped FOB destination.

Insert Table 1 About Here

Insert Table 2 About Here

All sales are on credit. Customers should receive an order confirmation on the web in response to their orders that includes a sequential order number that uniquely identifies each order, the customer's purchase order number, the products ordered, the quantity ordered, the cost of the individual items ordered, and the total cost of the order. You should focus on functionality and ensuring that the system meets information requirements. Specific issues to be considered in the analysis, design, and implementation of the system follow.

Analysis

In the analysis phase, examine company and customer needs. The analysis should identify issues that will be considered in designing and implementing the system. Some of these issues are:

- What information should be provided on the web site? This question should consider the information needs of customers about the company and its products and about using the web site.

- What data should be presented and obtained from the order page? This question should consider the information needs of customers.
- What information might be required by Al, shipping, and the accounts receivable clerk? This question should consider the control issues, as well as management information needs.
- What data should be included in the database? Database specifics are part of the design phase, but you should consider what data items will need to be stored in the database and where these data items will come from. Will the customer enter them? Are they provided by the database? Will they be generated by the system?

At the conclusion of this stage, you should have defined the use cases that describe a customer's typical interactions with AS when placing an order. You should also have developed a preliminary class diagram that depicts the objects required by AS to process orders from customers.

Conceptual Design

In the conceptual design stage, you should begin to identify how you will develop an e-commerce system to meet the needs of AS and its customers. You should plan for a three-tier architecture. Accordingly, the system will include creation of a web site that will be called by a customer's web browser. The browser will provide the client interface. The web site will run on a web server using ASP middleware to connect to a relational database system on a database server. Typically, the web server and database server would exist on separate computers and could even use different operating systems. They could reside on the same computer, however, and for purposes of this case, there is no particular reason that you need to focus on more than one platform.

The first major step of conceptual design is to finalize the database schema. Consider the structure of the relational database system you will recommend. Though variations are possible, the schema should include tables for maintaining customer and product data, for recording order data, and for relating the products and orders.

The next major step of conceptual design is to define the web pages that will comprise the e-commerce system, the function of each web page, and the relationships among the web pages in terms of how the customer will interact with the site. As a minimum, the site should provide a login page to restrict access to authorized customers only, a page that summarizes the current order, mechanisms for adding, updating, and removing items from the order, and a page that provides an order confirmation.

You should also consider any assumptions that you will make about the ways that customers will access the site, the equipment they will be using, and the web browsers they are likely to use. For example, if most of your customers will access the site via dial-up connections to ISPs (Internet Service Providers) instead of using high-speed broadband connections, this might influence your design in terms of the type and quantity of information presented on each web page.

Finally, you should consider security issues, including where SSL (Secure Socket Layer) should and should not be used, how access to the database should be controlled, and where security tools such as firewalls, virus scanners, and encryption should be utilized.

Once you have finalized your database schema and the overall structure of your web site, you are ready to proceed with the physical design and implementation of the system.

Physical Design and Implementation

In the physical design stage, you should consider implementation issues, including how your system will keep track of the customer's current order, where and what types of data validation to perform, the logical structure of the pages that require server-side processing, and how data will be exchanged between pages. Once you have completed the physical design, you should have a fairly detailed description of each page and the database. With this information you should construct the e-commerce system for AS.

As a first step in the physical design stage, you should decide how the site will maintain state information about the customer's order. With a web application, the web server does not remember any details of its prior interactions with the web browser. You must somehow provide a way for the e-commerce system to keep track of the items that are in the current order; this is state information. There are four major ways to maintain state information: using cookies, URL (Uniform Resource Locator) encoding, form fields, or a server-based "shopping cart" feature. You should consider each of these alternatives. Research and assess the advantages and disadvantages of each in terms of site usability, security, and complexity of the implementation.

Once you have determined how the site will maintain state information about the customer's current order, you should determine which page the customer will use to input each piece of information, as well as where and how the information must be validated. Generally, information can be validated either on the client or on the server. Where possible, information should be validated on the client to reduce the number of requests sent to the server. However, some types of data validation generally must be performed on the server (e.g., customer authentication). You should also consider ways to construct your pages to minimize the likelihood of customers entering invalid information.

Business logic, including interactions with the database, performing computations, and some data validation, is generally performed on the web server. With the ASP middleware, the business logic is implemented in scripts written using either the VBScript (Visual Basic Script) or JScript (a variant of Java) programming language. These scripts receive data from the client, process the data, and interact with the database using SQL commands. The scripts format and return data from the database to the browser. The scripts may include controls for verifying data before they are sent to the database. For each page that will require server-side processing, you should outline the logical structure of the script. You should consider the data passed to the script from the previous page, data validation steps to be performed, how state information is retained,

data to be retrieved from and written to the database, the conditions that may influence processing, and the information to be returned to the client.

You should determine how you will exchange data between the pages on the site. Passing data to the web server for processing is referred to as "submitting" the data to the server. There are three major ways to submit data to a web server: using cookies, URL encoding, or form fields. Once you have assessed the advantages and disadvantages of each, you should determine the best way to submit each information item captured on a page to the server for processing, given the requirements of the system and your experience.

Once you have completed the physical design of the order entry system, you should implement your physical design using ASP middleware and Microsoft Access. The process of implementing the system will involve constructing the database and web pages that comprise the system, entering customer and product data as described in Table 1 and Table 2, and testing the system by entering mock orders. Be sure to thoroughly test your system to ensure that data validation is performed and that orders are accurately stored in the database.

III. Teaching Notes

The teaching notes describe the use of the Al's Sounds case in an AIS course. As discussed earlier, this case is intended for an advanced AIS course and, as such, incorporates a number of advanced technological issues. These teaching notes include information about the technologies needed to use this case in an advanced AIS course, suggestions for preparing students to complete the case, challenges faced by students and instructors in using the case, guidelines for debugging ASP applications, and alternative approaches and technologies that could be used to implement the case. These teaching notes provide guidance for instructors who wish to use this case; however, individual instructors may wish to focus on various aspects of the case or expand the case to meet specific needs. Where appropriate, alternative technologies, web resources, and additional reference materials are discussed.

Technology Resources Needed

As a prerequisite for using this case in an AIS course, three technologies are required on the server side to implement Al's Sounds e-commerce site using ASP: a web server that supports ASP middleware such as Microsoft IIS (Internet Information Server) or Apache Web Server, ASP middleware such as Microsoft ASP or Sun Chili!Soft ASP, and a database management system such as Microsoft Access, Microsoft SQL Server, Oracle, or IBM DB2. Appendix A provides web addresses for these products.

For the sample implementation described in appendix E, Microsoft IIS, Microsoft's ASP middleware, and Microsoft Access were used. IIS is a full-featured web server capable of supporting small to large-scale web sites. IIS is currently available from Microsoft for most versions of Microsoft Windows with the exception of Microsoft Windows ME and Microsoft Windows XP Home Edition. ASP middleware is a script-oriented middleware technology that provides a number of resources for server-side processing. Most important for this case is the ASP middleware's robust data access support. Microsoft's ASP middleware is included with all current versions of IIS. Microsoft Access is a low-end RDBMS (relational database management system) capable of supporting a limited volume of transactions. For high volume sites, a more powerful RDBMS such as Microsoft SQL Server, Oracle, or IBM DB2 would be more appropriate.

The sample implementation is hosted on a Microsoft Windows NT 4.0 Server. In this setting it is important to install all Windows NT service patches and the most current versions of MDAC (Microsoft Data Access Components). These patches are available through Microsoft' web site.

A typical Internet application consists of a combination of web pages, programs or scripts, and a database. IIS and the ASP middleware support two primary types of web pages: HTML pages and ASP pages. HTML pages incorporate HTML and client-side scripts; however, no server-side processing is performed. ASP pages incorporate HTML,

client-side scripts, and server-side scripts that are processed by the ASP middleware to interact with databases and perform other processing actions.

Each student is provided with an account on the web server. As part of the account, each student has a personal home directory. The student's home directory is located in the web root directory (the directory where the web server looks for web pages). The student has Full Control permissions for this home directory, and the Everyone group has Read and Execute permissions. These permissions enable the student to place HTML and ASP pages in the home directory and then access these pages using a web browser. A subdirectory called `databases` is included in each student's home directory. The Everyone group has Change permissions for this directory so that the database can be accessed and updated via ASP pages. Additional information about web server security is provided below. Sample code is provided later to show how the connection between an ASP page and the database is defined. Students use FTP (File Transfer Protocol) to transfer files to their home directories. IIS provides server-side FTP support.

When constructing the web site, the technology resources needed by the student are minimal. The student needs access to a basic text editor like Microsoft Windows NotePad or Crimson Editor to create the web pages. The student also needs access to a database management system like Microsoft Access to construct the database. The student needs an FTP client to transfer the web pages and database to the web server; the academic edition of WS-FTP is an easy-to-use FTP client that is available for free to students. Finally, the student needs a current version of Internet Explorer to view the working web site.

Student Preparation and Resources

Our students prepared for this case by completing the Morrison and Morrison 2000 database-driven web sites (DDWS) tutorial. This tutorial provides knowledge about web-application concepts, general programming, SQL, HTML, and ASP programming using VBScript. Chapters one through four and eight are the most useful chapters for

preparing students to complete the case. Chapter one introduces web concepts. Chapter two introduces Microsoft Access and database concepts. Chapter three provides good coverage of data manipulation and data query commands supported by SQL. Chapter four provides an overview of HTML. Chapters one through four can be completed relatively quickly since the opportunity for students to make mistakes that take a long time to track down is relatively small; we allocated three weeks for students to complete these four chapters. Chapter eight integrates HTML, SQL, and VBScript code to create a database-driven web site. Chapter eight takes more time since most students are not used to programming or debugging errors in programs; as a result, we allocated two weeks for students to complete this chapter. In addition to completing the tutorial, students also read a number of articles published in professional and technology-oriented periodicals. These articles address design, implementation, and security issues relevant to the case. Appendix B provides a list of the articles. Upon completion of the tutorial and readings, the students work in groups of two to four students to complete the case. The case requires approximately three to four weeks to complete, depending on the backgrounds of the students, the amount of class time allocated to the case, and the time required to review materials submitted by students.

In addition to the tutorial, students have access to Childs, et al. 2000, Flanagan 1998, and Weissinger 2000, as well as the Internet. The Childs, et al. book provides extensive information about VBScript (Visual Basic Script) programming, syntax, and functions. This book serves as a useful reference when developing ASP scripts. The Flanagan book provides similar information for the JavaScript language and is useful when developing client-side functions. The Weissinger book summarizes the major features of the ASP middleware. Students are encouraged to use web search tools, including Google, Lycos, and Yahoo, to research issues identified in the case. Students also use ASP and HTML-oriented sites, including ASP 101, PowerASP, and the W3C (World Wide Web Consortium) site, as references when developing the application. Appendix A lists several web sites that provide ASP tutorials.

During the systems analysis and design stages of an information systems development project, models of systems requirements are produced so that end users, systems analysts, and developers can discuss the requirements of the system and ensure that the implemented system completely and correctly fulfills these requirements. UML is a popular notation for graphically representing the parts of a system and the interactions among those parts. The first version of UML was released in 1995. Since then, UML has emerged as a standard notation for modeling systems of all types. A number of companies have adopted UML as their standard modeling notation and a number of CASE tool developers have integrated UML support into their products. Because of its popularity, UML is a good modeling notation for people involved with the analysis, design, implementation, or auditing of information systems to know. Appendix C provides a brief introduction to UML. Students also can use various textbooks and online tutorials to develop a working knowledge of UML. Appendix A lists several web sites that provide UML tutorials.

Challenges Faced by Students and Instructors

The prerequisite knowledge required to implement even a rudimentary e-commerce system is significant. As a result, this case often represents a significant technological challenge for students and instructors alike. The following discussion describes the more common reactions and concerns of students to this case, as well as the steps that we have taken to address their reactions and concerns.

When starting on this case, students often ask the question, "Why are we doing this?" There are a number of possible answers, including the following. This case provides an opportunity for students to see the internal workings of a system, both in terms of how data are stored and how programs work. Exposure to these workings provides students with an understanding of the relationship between applications and databases. This case also enables students to understand the structure of systems, as well as general and application controls used to ensure the integrity of the data captured by a system. Finally, this case enables students to see some of the risks and challenges associated with e-commerce applications.

Even a short case with a limited number of requirements can present a significant challenge and leave students uncertain about how to get started. As a result, the case is divided into individual deliverables to match the major requirements: analysis, conceptual design, and physical design and implementation. Groups submit written reports and receive written feedback at each step. If time permits, groups present their work to the class, enabling students to learn from each another.

One of the greatest challenges faced by students attempting to build an Internet-based application for the first time is the integration of a diverse collection of technologies required to build the application. The procedural code executed on the server is implemented using a server-side scripting language (e.g., VBScript or JScript). Queries to interact with the database are implemented using SQL. The information to be presented to the user by the web browser must be marked up using HTML. Finally, data validation procedures performed by the browser are implemented using a client-side scripting language (e.g., VBScript or JavaScript). Students must develop the skills to use the various languages.

Rather than developing these skills simultaneously, students are introduced to each of the languages as they work through the Morrison and Morrison 2000 tutorial. After the students finish a chapter in the tutorial, they complete a structured exercise that covers the language addressed in the chapter. The exercises enable students to practice with the language and develop a greater level of comfort before they have to apply their knowledge to complete the case. To reduce the number of scripting languages required of students, VBScript is used for both server-side and client-side programming. This approach also simplifies the coding task since VBScript is not case sensitive, thus reducing a common source of errors. Finally, we provide source code for common data validation and formatting routines to the students, permitting the students to focus on when to use these routines rather than how to implement them.

One challenge either the instructor or a network administrator faces is configuring the web server to prevent unauthorized access and misuse, while at the same time providing students with the ability to create and maintain their web sites. Web-based applications involve two related, but different, types of security: web server security and web site security. Web server security deals with configuring the web server to minimize the risk of cyber-criminals gaining access to and control of the web server. Web site security deals with designing and implementing a web site to minimize the risk of cyber-criminals entering fraudulent transactions, disrupting operations, destroying data, or gaining access to sensitive data in the database. The first part of appendix D addresses securing an IIS web server to prevent known IIS exploits from working. The remainder of appendix D addresses general e-commerce security issues.

One final challenge both students and instructors face is debugging the code produced while working through the DDWS tutorial and developing the ASP application for Al's Sounds. The next section provides some guidelines for tracking down errors in ASP code.

Debugging ASP Applications

Once groups begin developing the web pages, scripts, and queries to implement the case, they are likely to encounter programming problems. Unfortunately, the error messages generated by web browsers, web servers, and the ASP middleware are often difficult to interpret. Debugging an Internet application can test the skills of both students and instructors. Fortunately, certain types of errors occur frequently, including the following.

- The wrong delimiter is used to open or close a script, or the delimiter is not provided. ASP scripts are placed between an opening tag of `<%` and a closing tag of `%>`. Client-side scripts are placed between an opening tag of `<script …>` and a closing tag of `</script>`. When the proper script delimiter is omitted, the web browser generally displays script code as part of the web page, instead of executing the script.

- A language keyword is misspelled. This error triggers a syntax error message that includes a fragment of the code containing the misspelled keyword.
- A variable name is misspelled. This error can be difficult to track down unless the ASP middleware is instructed to verify that each variable is declared before being used. The first server-side instruction in each ASP page should be `Option Explicit`. This instruction tells the ASP middleware that each variable must be declared prior to use. If a variable name is misspelled, the ASP middleware generates a runtime error that displays the undeclared variable name.
- A variable is assigned an invalid value based on the type of data that the variable is supposed to contain or the operations to be performed using the variable (a type mismatch). For example, if a string value is assigned to a variable (e.g., anum="ABC") and then an attempt is made to perform a mathematical operation using the variable (e.g., anum = anum + 1), a type mismatch error occurs. The ASP middleware generates an error message displaying the incorrect value and the line number where the error was detected.
- An invalid database table name or field name is used in an SQL query or ASP code. This type of error typically results in either a database engine error or a recordset error. A line number is displayed indicating the line in the ASP page where the error was detected. This line number can be deceptive because the actual error may occur earlier in the code than the location where it is detected. For example, if an SQL query stored in a variable contains an invalid field name, the error is not detected until the script attempts to execute the query using the variable.

When debugging an ASP page, it is important to remember that the line number displayed with an error message indicates where the error is detected, not necessarily where the error exists in the source code. If the error is detected by the ASP middleware, the line number points to a location in the original ASP page source. If the web browser detects the error, the line number points to a location in the web page produced by the ASP middleware, not the ASP page source itself. As a result, it is helpful to look for other cues to point to where the error occurred including fragments of code, variable names, or values displayed as part of the error message. Finally, a text editor that

displays line numbers can be helpful in tracking down errors. Some versions of Microsoft Windows NotePad can be configured to display line numbers. Crimson Editor, a free text editor, can also display line numbers. Crimson Editor also provides a language syntax feature that displays language keywords in different colors; this can simplify debugging significantly.

Two useful resources when designing and building web applications are Microsoft's Developer Network (MSDN) and Internet search tools such as Google. A number of different MSDN subscriptions are available from Microsoft for a fee; however, many of the articles that relate to ASP, VBScript, and Internet application development are also available for free through Microsoft's web site. Microsoft's web site also includes a search feature that enables users to locate articles and pages on the web site. However, more general Internet search tools often provide access to the same information, as well as information on non-Microsoft web sites. Developers and consultants sometimes provide better descriptions for errors and easier ways to work around errors than might be available on the Microsoft web site. When faced with an error that eludes a solution, a good strategy is to search both Microsoft's web site and the Internet in general.

Alternate Instructional and Implementation Approaches

In addition to having students complete the entire case from analysis through implementation using ASP, instructors may choose to use the case in different ways depending on the students' backgrounds and the resources available. One approach would be for the students to complete part of the case, stopping after analysis, conceptual design, or physical design, depending on the focus of the course. As a second approach, alternate middleware technologies, including JSP (Java Server Pages), PHP Hypertext Preprocessor, MacroMedia ColdFusion, or PERL (Practical Extraction and Report Language), could be used during the physical design and implementation stage with minimal changes to the case. Although the sample system described in appendix E was hand-coded using a basic text editor, a third approach would be for students to use an IDE (Integrated Development Environment) tool like Microsoft FrontPage or

MacroMedia DreamWeaver UltraDev during the implementation. These tools enable students to build web applications with minimal programming knowledge. A disadvantage of these tools for educational purposes is that they can conceal the logical processes that are required to create meaningful business applications.

Summary

The Al's Sounds case provides students with an opportunity to practice systems analysis, design, and implementation skills in an e-commerce setting. As discussed in these teaching notes, significant technological resources are required to implement the case. We suggest using IIS and provide recommendations for configuring the web server and student accounts. We also describe alternate web server technologies that can be used. Because of the advanced technologies used in this case, preparation is critical for ensuring that students are successful in completing this case. We suggest using the Morrison and Morrison 2000 tutorial, coupled with in-class exercises, to prepare students. Students and instructors using this case will face a number of challenges. We summarize these challenges, and provide suggestions for responding to these challenges. One significant challenge for both students and instructors is debugging ASP applications. We provide some recommendations for finding the more common errors made by students when writing ASP scripts. Finally, we provide information about alternate teaching approaches that instructors can use with this case.

Table 1

Sample Customer Data for Al's Sounds

CustomerID	Name	ShipStreet	ShipCity	ShipState	ShipZip
1	Indie Music	4 Reeves Blvd.	Dover	OH	44683-
2	CD's Today	1871 Hardy	Hattiesbur	MS	39401-
3	University	1531 High	Columbus	OH	43211-
4	OriginL	187 Sawmill Rd	Columbus	OH	43209-
5	D&F Music	101 Fourth St	New Phila	OH	44663-

CustomerID	BillStreet	BillCity	BillState	BillZip	Phone
1					(555) 555-
2	18 Hardy	Hattiesburg	MS	39402-	(555) 555-
3					(555) 555-
4					(555) 555-
5	8973 72nd St	New Phila	OH	44663-	(555) 555-

CustomerID	Login	Password
1	im	pqr@320
2	cdt	tuv@911
3	ut	asd@665
4	ols	bwl@423
5	dfm	ojo@908

Table 2

Product Data for Al's Sounds

ProductID	Title	Artist	Year	ListPrice
1	Bill Blues Greatest Hits	Bill Blue	1982	$19.95
2	Live at the Metro Station	Subway Willy	1975	$12.99
3	I Like Cheese	Eddie and the	1984	$9.98
4	Love Songs at the Movies	Multiple	1990	$14.95
5	Crickets in the Grass	Downtown Strollers	1973	$13.99
6	CJs Dance Tunes	CJ McCall	1984	$15.99
7	Rolling Moss Gather No	UC and Lucy	1993	$12.95
8	Camp Songs	Subway Willy	1978	$11.95
9	Guitar Man	Willy Cole	1981	$21.99
10	Dancing King	Babba	1979	$15.95

Appendix A

Web References

Resource	Address
Case Materials	
Al's Sounds Demonstration and Source Files	demos.is-doctor.com
Application Development Tools	
MacroMedia ColdFusion MacroMedia DreamWeaver UltraDev	www.macromedia.com
Microsoft FrontPage	www.microsoft.com
ASP Middleware and Web Servers	
Apache Web Server	www.apache.org
Microsoft IIS and ASP	www.microsoft.com
Sun Chili!Soft ASP	www.chilisoft.com
Database Management Systems	
IBM DB2	www.ibm.com
Microsoft Access Microsoft SQL Server	www.microsoft.com
Oracle	www.oracle.com
Other Middleware Technologies	
PERL (Practical Extraction and Report Language)	www.perl.org
PHP Hypertext Preprocessor	www.php.net
Search Sites	
Google	www.google.com
Lycos	www.lycos.com
Yahoo	www.yahoo.com
Security Information	
CERT	www.cert.org
McAfee AVERT	www.nai.com
Netscape's SSL Information	netscape.com/security/techbriefs/ssl.html
Symantec Security Response	www.symantec.com
Training and References	
ASP 101	www.asp101.com
Object Mentor (UML tutorials)	www.objectmentor.com
PowerASP	www.powerasp.com
Sparx Systems (UML tutorial)	www.sparxsystems.com.au
W3C (World Wide Web Consortium)	www.w3.org
W3Schools.com (HTML, ASP, VBScript, SQL, …)	www.w3schools.com
Webopedia	www.webopedia.com
Utility Programs	
Crimson Editor	www.crimsoneditor.com
WS-FTP	www.wsftp.com

Appendix B

Suggested Readings

Databases

Hayes, D. C. and J. E. Hunton. 1999. What You Better Know About Databases. *Journal of Accountancy* (January): 61-63.

Hayes, D. C. and J. E. Hunton. 1999. Building a Database from Scratch. *Journal of Accountancy* (November): 63-73.

Hayes, D. C. and J. E. Hunton. 2000. Working with Databases. *Journal of Accountancy* (May): 70-79.

Hayes, D. C. and J. E. Hunton. 2001. When Querying Databases, You've Got to Ask the Right Question. *Journal of Accountancy* (February): 35-45.

Hayes, D. C. and J. E. Hunton. 2002. Put a Database to Work. *Journal of Accountancy* (January): 69-76.

Strehlo, C. 1988. Mastering the Shared Database. *Personal Computing* (December): 121-125.

Wolfe, C. and S. E. Yoder. 1986. Micros in Accounting: Designing Databases for Accounting. *Journal of Accountancy* (October): 138; 140; 142-143.

Security

DeMaio, H. 1999. Getting Started With PKI. *Information Systems Security* (Summer): 27-31.

Harrison, A. 2000. Digital Certificates. *ComputerWorld* (August 14): 58.

Moscove, S. A. 2001. E-business Security and Controls. *The CPA Journal* (November): 40-46.

Newing, R. 1997. Secure Internet Transactions At Last! *Management Accounting* (March): 44-45.

Punch, L. 2000. Internet Fraud's New Threat. *Credit Card Management* (April): 84-92.

Radcliff, D. 1998. Is Your ISP Secure? *InfoWorld* (March 2): 97; 100.

Walsh, B. 1997. Internet Security: A Technical Report. *Texas Banking* (May): 16-17.

Web Applications

Geerts, G. L., B. A. Waddington, and S. C. Dilley. 2000. Sophisticated Web Design. *Journal of Accountancy* (April): 55-59.

Radding, A. 1999. Overcome the Web-Transaction Barrier. *Information Week* (December 13): 153-154; 158; 160.

Appendix C
Unified Modeling Language Overview

During the systems analysis and design stages of an information systems development project, models of systems requirements are produced so that end users, systems analysts, and developers can discuss the requirements of the system and ensure that the implemented system completely and correctly fulfills these requirements. UML is a popular notation for graphically representing the parts of a system and the interactions among those parts. The first version of UML was released in 1995. Since then, UML has emerged as a standard notation for modeling systems of all types. A number of companies have adopted UML as their standard modeling notation and a number of CASE tool developers have integrated UML support into their products. Because of its popularity, UML is a good modeling notation for people involved with the analysis, design, implementation, or auditing of information systems to know. This section provides a brief introduction to UML.

UML uses an object-oriented approach. It treats a system as a set of objects. Objects are capable of containing data and performing operations. Objects that comprise a system interact in response to users' requests. Each object is a member or instance of a class. All objects in a class share common features, such as operations and data items.

UML provides graphical representations of the objects and classes that comprise a system and the behaviors exhibited by the system. The purpose of UML is to help systems developers understand the needs of users and the required functionality of a system and to translate those needs into the components of a working system. At a minimum, use cases and class diagrams are produced when using UML. Additional behavior and implementation models are produced based on the requirements of the system being modeled. For examples, see Fowler and Scott 1998 and Booch, et. al. 1999.

A use case depicts an interaction between a user and the system (Fowler and Scott 1998). Each different interaction with the system is represented as a separate use case.

See Figure 1 for a general example of a use case. The use case includes an actor who represents the role that the user plays in relation to the system. The actor generally initiates a use case. The use case depicts the primary flow of events performed by a system in response to a request from the actor. Accordingly, it is a useful tool for modeling the way a particular user interacts with a system and the types of actions the system will perform in response to user requests.

Insert Figure 1 About Here

Users are identified in the diagram by a stick figure. Examples of users are customers, account managers, shipping clerks, and sales representatives. Actions are represented by ovals, containing labels that identify the action. Examples of actions include login, ordering, updating, and shipping. Lines link the user with each action. Additional components of a use case can extend its functionality. For examples, see Schneider and Winters 1998.

A class diagram describes the classes of objects that comprise a system and the relationships among the classes. See Figure 2 for an example of a class diagram. Each class consists of a title for the class, attributes that represent the relevant facts about the objects that comprise the class and the operations that the objects can perform. The class diagram is similar to an entity relationship diagram, especially when applied to modeling a relational database. Tables in the database can be represented as classes. Fields in the table are listed as class attributes. Operations identify programmed procedures that the system can perform involving data in the table. A class diagram can contain other components that extend its functionality. See Fowler and Scott 1998.

Insert Figure 2 About Here

An example of a class might be a customer. Name and address could be attributes, and validateLogin() might be an operation. Each customer is a particular object in the class Customers, and each shares the same set of attributes and operations

within that class. The format of the diagram varies a bit in practice. Typically, a box represents each class. The name of the class is listed in the first section of the box. Attributes are listed in the second section. Attribute names begin with lower case letters followed by a colon. The data type of the attribute (string, integer, currency, date) follows the colon. Operations are listed in the third section. Operations are followed by parentheses and implemented as programming code. Lines separate the sections. Lines also are used to connect related classes as in an entity-relation diagram.

Figure 3 provides a sample use case, and Figure 4 provides a class diagram for the AS case.

Insert Figure 3 About Here

Insert Figure 4 About Here

Figure 1

Example of a Use Case Diagram

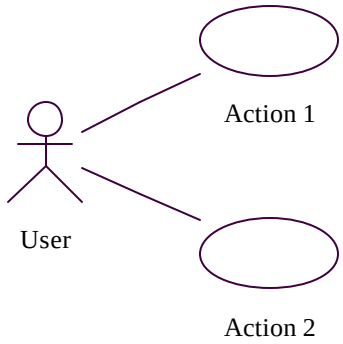


Figure 2

Example of a Class Diagram

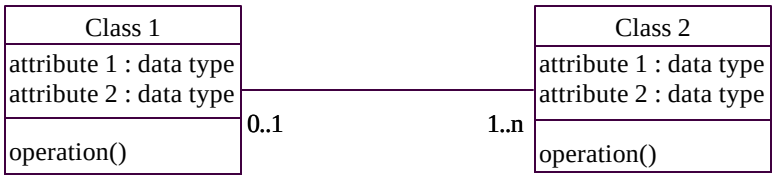


Figure 3

Use Case Diagram for Customer

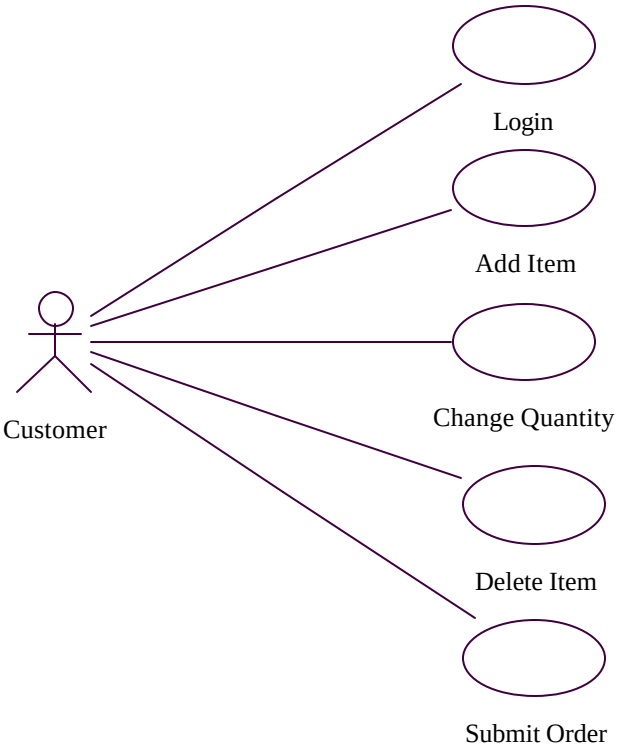
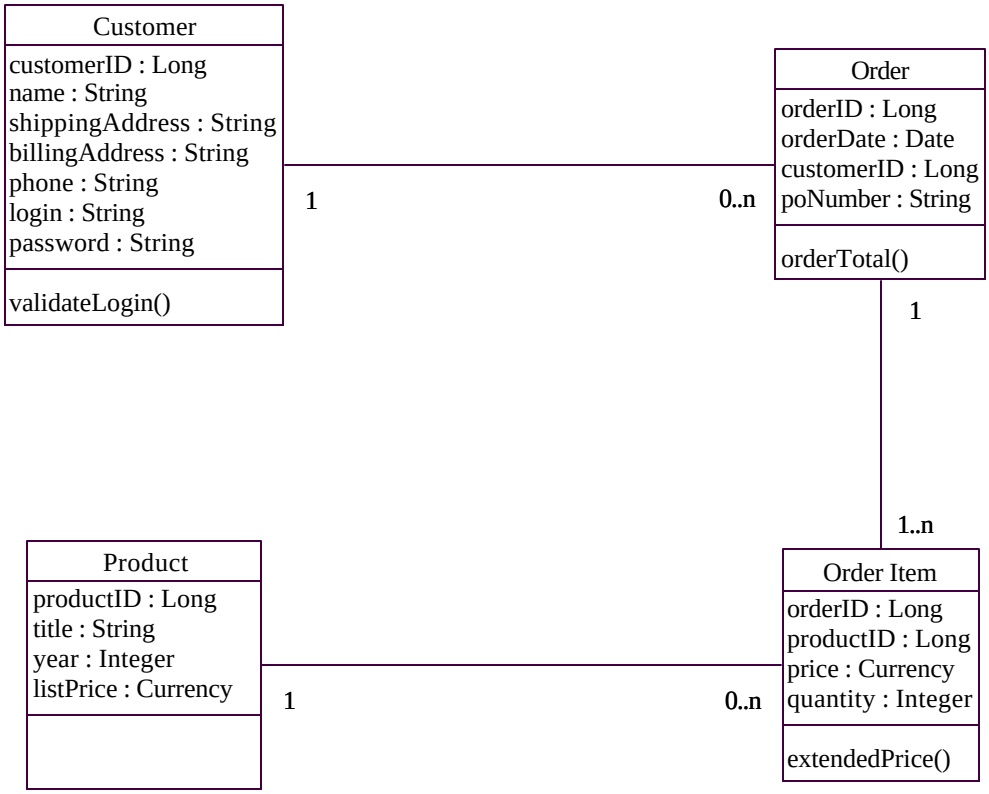


Figure 4

Class Diagram for Al's Sounds Database



Appendix D

Security Issues

Web-based applications involve two related, but different, types of security: web server security and web site security. Web server security deals with configuring the web server to minimize the risk of cyber-criminals gaining access to and control of the web server. This is the most basic level of security that needs to be provided. Web site security deals with designing and implementing a web site to minimize the risk of cyber-criminals entering fraudulent transactions, disrupting operations, destroying data, or gaining access to sensitive data in the database. This section discusses these two security issues.

Web Server Security

Web servers need to be configured so authorized users can access resources, while preventing unauthorized access. This configuration requires a careful balancing of permissions and options. The following recommendations are intended to help reduce the likelihood of a successful attack on an IIS web server. These recommendations apply to instructional and e-commerce settings.

1. Disable Anonymous FTP access.
2. Remove all virtual directories installed by IIS. The removal eliminates web-based administration; however, many attacks on IIS web servers use these virtual directories.
3. Disable directory browsing. This prevents users from seeing directory lists when they access the server using a web browser; however, it also prevents attackers from easily determining the structure of the students' home directories.
4. Do not install the Posting Acceptor add-on. This tool is often used to plant backdoors and zombies on web servers.
5. Provide the Everyone group with Read and Execute permissions only on the web root directory and students' home directories.
6. Monitor Microsoft security bulletins and apply IIS security patches when released.

7. Monitor security sites such as CERT, Symantec Security Response, and McAfee AVERT for security alerts and take preventive measures as recommended.
8. Apply Microsoft Windows updates regularly.
9. Monitor the IIS logs for signs of suspicious activities and adjust security settings as required.
10. Use a virus scanner on the web server and maintain current virus signatures.

Web Site Security

Web site security is a critical issue for any e-commerce company or business providing either transaction-processing services or access to sensitive information online. Security experts and cyber-criminals discover new attack methods and weaknesses in the systems hosting Internet applications almost daily. As a result, web site security practices are constantly evolving. A web site security policy must address a variety of attacks including attacks directed at gaining unauthorized access to information traveling through the Internet, denial-of-service (DOS) attacks, misappropriation or destruction of server resources, and unauthorized access to the database maintained by the web site. The following are general guidelines for securing a web site from attacks.

To protect against attacks directed at gaining unauthorized access to information traveling through the Internet, SSL is used when sensitive information is transmitted. As a minimum, the web server must have a digital certificate that identifies the web server and contains a public encryption key. When the user is directed to a web page that requires a secure connection between the web browser and web server, SSL establishes the secure connection using the server's digital certificate. Once the secure connection is established, the web browser and web server can transmit encrypted information to ensure confidentiality and message integrity. When using SSL, the trade-off between confidentiality and performance must be assessed. SSL requires significantly more server resources to encrypt and decrypt transmissions; as a result, excessive use of SSL can decrease server performance.

DOS attacks directed at preventing authorized users from accessing a web site render the web site inoperable by either blocking the path to the web site or overloading the web server so that it cannot respond to legitimate requests. Protection against DOS attacks starts at the ISP or Internet backbone provider's facility. The routers used on the Internet backbone and by ISPs can be configured to block known DOS attacks from being permitted to propagate through ISP networks down to the lower-bandwidth connections linking individual companies to the Internet. By filtering attacks at points closer to the Internet backbone, where high bandwidth connections are used, the impact of the DOS attack is reduced. To protect against DOS attacks that exploit known defects in web server software, the software must be updated or patched to repair the defect. As a result, procedures must be implemented to regularly apply service patches on web servers.

In a number of cases, cyber-criminals have managed to penetrate and misappropriate or destroy server resources. The goal of the cyber-criminal may be to deface a web site, use a server to host a pirated software distribution site (warez), alter or destroy files, or otherwise misuse resources. The consequences of this type of attack range from simple embarrassment to the destruction of critical business information. There are a number of precautions that should be taken to reduce the likelihood of cyber-criminals accessing servers. As discussed earlier, each web server should be configured to minimize exposure by removing unneeded features, applying updates and security patches, disabling anonymous access, and setting appropriate permissions. Additionally, a firewall should be placed between the web site and the Internet, and the firewall should be configured to block all access using FTP, Telnet, and other Internet application protocols not specifically required.

Perhaps the computer crime that has been most underestimated, but offers the most potential for damage to an e-commerce company and its customers is unauthorized access to the e-commerce company's database. In the last few years reports of successful penetration attacks aimed at giving the cyber-criminal access to an e-commerce company's database have become common. As a result, a number of security measures

have evolved to address this threat. As already discussed, the connection between the Internet and the e-commerce site should be protected by a properly configured firewall that permits only specific types of access to the e-commerce site. On the web server, whenever possible, permissions should be set so that scripts can only be executed, not read. This feature is not available with all web servers; for example, it is available with IIS 5 but not IIS 4. The database should be installed on a different computer than the web server, and the two computers should be connected using a LAN that does not use TCP/IP. This makes unauthorized access to the database more difficult since the cyber-criminal must first gain access to the web server before the cyber-criminal can attack the database server through the LAN. The database should be password protected and permissions should restrict access to the various tables and field. Finally, inside the database, sensitive information should be encrypted so that even if the cyber-criminal manages to gain access to the database, the information is still protected.

One additional security issue relates to how information is exchanged between the web browser and the web server. As discussed below, there are three major ways to exchange data between the web browser and web server: cookies, URL encoding (embedding data in the address of the requested web page when issuing a request to a web server), and form fields. In selecting the method for exchanging data, several security issues arise. Critical security issues include:

- Can others see the data on the screen during the user's session?
- Does the data get saved in the web browser's history?
- Does the data appear in the web server's log?

In general, URL encoding is not appropriate for the exchange of sensitive information since the information can be seen in the web browser's address box, is saved in the web browser's history, and appears in the web server's log.

Appendix E

Sample Implementation of the Al's Sounds Case

This section describes a sample implementation of the Al's Sounds case. The sample implementation consists of one HTML page (Login.html) and three ASP pages (OrderItems.asp, AddItem.asp, and SubmitOrder.asp). Figure 5 depicts the logical relationship among the web pages. This sample implementation uses a Microsoft Access database (AS.mdb) to store data about customers, orders, and products. The sample implementation, outlines of the logical structure of the scripts, and source code are available at demos.is-doctor.com.

Insert Figure 5 About Here

As discussed in the case, two critical physical design issues must be addressed when building an e-commerce site: how will data be exchanged between the various pages, and where will data associated with orders in progress be stored? Three methods can be used for exchanging and storing data until an order is completed: cookies, URL encoding, and form fields. All of these methods employ a name/value pair, where each piece of data is assigned a name and a value is associated with the name (e.g., customerid=jdoe). The page passing the data is responsible for linking the value to the name. The page receiving the data is responsible for retrieving the value using the same name. ASP middleware provides built-in server-side support for all three methods. Each of these methods has advantages and disadvantages.

A chief advantage of using cookies is that, once a cookie has been set, it is available from any web page on the current site. There are several disadvantages of cookies, including some users disable cookie support in the browser, resulting in web pages not working properly, the amount of data that can be passed using cookies is limited, and client-side access to cookies requires sophisticated programming skills.

The chief advantage of using URL encoding is that the data being passed between web pages is saved to the browser's history list so a web page can be easily accessed later using the same data. This advantage turns out also to be a serious disadvantage for URL encoding; anyone with access to the browser or the server's log files can easily see the data, potentially compromising security. Additional disadvantages are that client-side URL encoding requires sophisticated programming skills and the amount of data that can be passed is limited.

Use of form fields offers several advantages, including ease of programming the client-side of the application using standard HTML form elements, virtually an unlimited amount of data can be exchanged, and security is enhanced if the Post method is used when submitting the form data to the web server. If the Post method is used, the data do not appear in the URL of the requested page, and the data are not written to the server's log files. Also, when used, SSL encrypts form data so that cyber-criminals cannot understand the data even if they capture it using a packet sniffer. The chief disadvantage of using form fields is that any data to be exchanged must be stored in a form input field that is part of the form actually submitted to the web server. A single web page may include multiple forms; however, only one form can be submitted to the web server at a time. To compensate, HTML provides the capability to define hidden form fields, enabling a form to contain large quantities of data that the user does not actually see.

In addition to storing data using the three methods discussed above, data associated with an order in progress can be stored in a "shopping cart" feature on the server until the order is completed. This method is especially useful if the customer is not able to complete the order in one visit to the web site and would like to return to the same point in the order process at a later time. At the same time, this method is difficult to implement because it requires a more complex database design and update procedures, and procedures must be defined for detecting and canceling dormant orders.

The authors' implementation uses a combination of cookies and form fields to exchange and store data. A single cookie is used to store the internal customer ID of the

customer currently accessing Al's Sounds. This approach is useful because any page may need the internal customer ID, and using a cookie eliminates the need for each page to maintain the internal customer ID. Also, the cookie provides an easy way to determine if the user has successfully logged on to the system. Our implementation uses form fields to exchange data between web pages and to store the products currently in the order. Only when the customer requests that the order be submitted for processing does data actually get written to the database.

AS.mdb

The Al's Sounds database consists of four tables: Customers, Orders, OrderItems, and Products as depicted in Figure 4.

A critical question associated with the database used by an e-commerce system is where to locate the database. Generally the database should be located on a separate server from the web server so that cyber-criminals who gain access to the web server will not automatically have access to the database containing sensitive or confidential data. In a classroom setting, this level of security is not necessary. As discussed earlier, this sample implementation assumes that the database is stored in a "databases" subdirectory of the student's home directory. This approach enables each student to work with an individual database when building and testing the web site.

The ASP middleware generally requires a fully qualified physical pathname to the database. Fortunately ASP provides an easy way to determine this physical pathname. Given a relative pathname, the MapPath function of the ASP Server object generates a fully qualified physical pathname to any file within the web server's root directory. The general syntax of the MapPath function is:

```
Server.MapPath("relative path from ASP page's location")
```

As an example, assume the web server's root directory, where student accounts are located, is `d:\users`, the ASP pages are located in the `jdoe` subdirectory, and the

database is located in the `databases` subdirectory. When executed by the ASP middleware, the following code creates the physical pathname `d:\users\jdoe\databases\as.mdb`:

```
Server.MapPath("databases\as.mdb")
```

The sample implementation uses this technique to determine the location of the Al's Sounds database. The following code uses the MapPath function to open a database:

```
connDB.Open "Driver={Microsoft Access Driver (*.mdb)};" & _
            "DBQ=" & Server.MapPath("databases\as.mdb")
```

Login.html

Login.html displays a form. When the user clicks the Login button on the form, Login.html verifies that the user has entered values for the customer ID and password fields. If so, Login.html submits the form to the server for processing. Otherwise, Login.html displays an error message and waits for the user to enter a customer ID and password.

OrderItems.asp

OrderItems.asp validates the customer ID and password. If the customer ID and password are not valid, OrderItems.asp displays an error message and redirects the user back to Login.html. If the customer ID and password are valid, OrderItems.asp displays a form with customer and order information. OrderItems.asp allows the user to add items to the order, change the quantity ordered, and delete items from the order. If the user clicks the Add Item button, OrderItems.asp submits a request to the server for AddItems.asp. If the user changes the quantity of an item and clicks the Update button, OrderItems.asp is executed again to re-compute the extended price and order total. If the user clicks the Delete button for an item, the order quantity is set to zero and OrderItems.asp is executed again to remove the item from the order and re-compute the

order total. If the user clicks the Submit Order button, OrderItems.asp submits a request to the server for SubmitOrder.asp.

AddItem.asp

AddItems.asp enables the user to add new items to the order. The user selects an item from the list displayed and then specifies the quantity to be ordered. When the user clicks the Add Item button, the new item is added to the order and OrderItems.asp is executed again.

SubmitOrder.asp

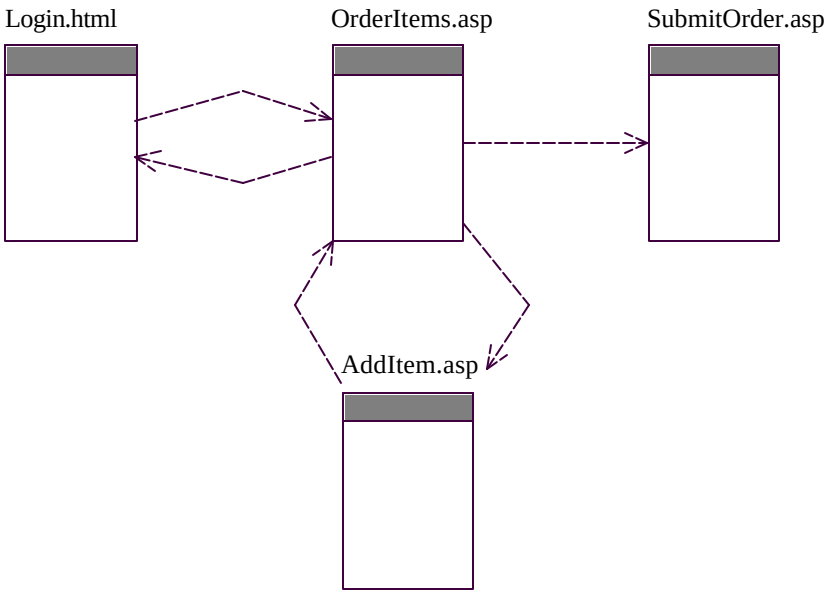
SubmitOrder.asp saves the order information to the database and displays an order confirmation with the order number, customer information, order item information, and order total.

Possible Extensions to the Al's Sounds Sample Implementation

There are a number of possible extensions to this sample implementation, including using database transaction when saving the order and providing a "shopping cart" feature. As currently implemented, SubmitOrder.asp contains a fairly significant flaw. If a failure occurs during the server-side processing, there is the possibility for a partial order to be saved to the database. Implementing database transaction processing code to test for errors and rollback the partial order if any errors occur easily solves this problem. This extension would illustrate a fundamental requirement of accounting systems: all updates related to a transaction should be processed successfully or none of them should be retained. As discussed earlier, adding a "shopping cart" feature would enable users to build an order over the course of several visits to the web site, rather than having to complete the order in one visit. This approach introduces a number of interesting technical and procedural challenges that must be addressed, including how to detect and cancel dormant orders. This extension would help to illustrate the relationship between policies and the implementation of an e-commerce site, including how long incomplete orders should be retained, how to secure access to orders-in-progress, and when to reduce inventory for items being ordered.

Figure 5

Sample Site Structure



References

Booch, G., J. Rumbaugh, and I. Jacobson. 1999. *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley.

Childs, M., P.Lomax, and R. Petrusha. 2000. *VBScript in a Nutshell*. Sebastopol, CA: O'Reilly.

Flanagan, D. 1998. *JavaScript: The Definitive Guide*. Sebastopol, CA: O'Reilly.

Fowler, M. and K. Scott. 1998. *UML Distilled: Applying the Standard Object Modeling Language*. Reading, MA: Addison-Wesley.

Morrison, M. and J. Morrison. 2000. *Database-driven Web Sites*. Cambridge, MA: Course Technology.

Schneider, G. and J. P.Winters. 1998. *Applying Use Cases: A Practical Guide*. Reading, MA: Addison-Wesley.

Weissinger, A. K. 2000. *ASP in a Nutshell*. Sebastopol, CA: O'Reilly.