Web-based Commerce
Auto Parts Store

By

Victor Hsu

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
OMI College of Applied Science

March 2000
Web-based Commerce
Auto Parts Store

By

Victor Hsu

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

Copyright 2000 Victor Hsu

The author grants the Information Engineering Technology Program permission to reproduce and distribute copies of this document in whole or in part.
Table of Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Content</td>
<td>i</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Review of Literature</td>
<td>1</td>
</tr>
<tr>
<td>3. Description of Solution</td>
<td>2</td>
</tr>
<tr>
<td>3.1 User Profile</td>
<td>2</td>
</tr>
<tr>
<td>3.2 Design Protocol</td>
<td>3</td>
</tr>
<tr>
<td>4. Deliverables</td>
<td>3</td>
</tr>
<tr>
<td>5. Design and Development</td>
<td>4</td>
</tr>
<tr>
<td>5.1 Budget</td>
<td>4</td>
</tr>
<tr>
<td>5.2 Timeline (tasks and schedule)</td>
<td>5</td>
</tr>
<tr>
<td>5.3 Software</td>
<td>6</td>
</tr>
<tr>
<td>5.4 Hardware</td>
<td>6</td>
</tr>
<tr>
<td>6. Proof of Design</td>
<td>7</td>
</tr>
<tr>
<td>6.1 Data Architecture</td>
<td>7</td>
</tr>
<tr>
<td>6.1.1 User Data</td>
<td>7</td>
</tr>
<tr>
<td>6.1.2 Parts Data</td>
<td>8</td>
</tr>
<tr>
<td>6.1.3 Order Data</td>
<td>9</td>
</tr>
<tr>
<td>6.2 Application Architecture</td>
<td>12</td>
</tr>
<tr>
<td>6.2.1 Implementing Product Search</td>
<td>12</td>
</tr>
<tr>
<td>6.2.2 Implementing Shopping cart</td>
<td>15</td>
</tr>
<tr>
<td>6.2.3 Login/Registration</td>
<td>16</td>
</tr>
<tr>
<td>6.2.4 Checkout</td>
<td>18</td>
</tr>
<tr>
<td>6.3 Web Layout</td>
<td>20</td>
</tr>
<tr>
<td>6.4 Data Entry Application</td>
<td>21</td>
</tr>
<tr>
<td>7. Conclusion and Recommendations</td>
<td>23</td>
</tr>
</tbody>
</table>

Appendices

- Appendix A: Stored Procedure
- Appendix B: Add Item to Shopcart
- Appendix C: Delete Item from Shopcart
- Appendix D: Clear all Item from Shopcart
- Appendix E: Checkout2.asp
- Appendix F: Data Entry Application Source Code


List of Figures

Figure 1. Timeline 5
Figure 2. IET Lab Servers 6
Figure 3. Design View of the Customer Table 7
Figure 4. Design View of the Parts Table 8
Figure 5. Design View of the Vehicles Table 9
Figure 6. Design View of the VehicleParts Table 9
Figure 7. Design View of tbl_shopcart Table 10
Figure 8. Design View of Orders Table 11
Figure 9. Design View of OrderDetail Table 11
Figure 10. Part Search by Vehicle 13
Figure 11. Search Results 14
Figure 12. Parts Detail 15
Figure 13. Shopping Cart 16
Figure 14. Login Page 16
Figure 15. New Account Registration 17
Figure 16. Registered Information 18
Figure 17. Checkout 1 19
Figure 18. Checkout 2 20
Figure 19. Basic Layout of the Site 20
Figure 20. Data Entry Application 21
Figure 20.1 Vehicle and Parts Data Entry Form 21
Figure 20.2 Parts and Vehicle Part Data Entry Form 22
Abstract

This report covers the use of the Internet as a new sales channel for brick and mortar stores. The purpose of this project is to use e-commerce to market as well as sell products for Cincinnati Generators and Starters, an automotive parts supplier. Interface with the database, credit card payment, user interface, and Internet security issues were considered in development phase. The material covered demonstrates what it takes to develop an e-commerce Web site. Data architecture, application architecture, and the web layout are discussed in details.

It makes good business sense to have a Web site even if it serves as an online catalog or marketing presence. E-commerce allows store to market and sell nationally. The effectiveness of the online Auto-parts Store could not be measured because this is a test project to show how e-commerce can help to market and sell products online.
Web-based Commerce – Auto Parts Store

1. Introduction

Cincinnati Generator and Starter (CGS) has been selling rebuilt/reconditioned starters and alternators for the past ten years in greater Cincinnati. Its main customers include nearby garages, transportation services, and walk-in customers. In the past couple of years the family-owned business has seen steady sales to its customers and would like to expand its services throughout the states. As a small business, it does not have sales force or the revenue needed to market its product through traditional means.

The owner had talked about selling its products on the Internet a year ago but at the time it seemed distant; electronic commerce was for the large corporations that could spend millions of dollars. Since that time e-commerce has become affordable for small businesses. The World Wide Web provides a new sales channel for those that have traditionally relied on a sales force for direct and indirect sales.

2. Review of Literature

Marketing on the Internet offers potential to achieve increased revenues and profits. The Internet can extend a company’s reach significantly enabling it to do business with new customer base that was previously in accessible. Online marketing is highly cost effective. Net marketing costs are significantly less than producing TV commercials or traditional print campaigns. With net marketing you can target consumer groups by e-mail. With targeting the interactive level of the customer is much higher because they have already indicated an interest in the company’s product/service.

Companies that do not develop an Internet presence risk loosing their customers to more aggressive competitors. It makes good business sense to have a Web site
even if all you have is special announcements. It can allow companies to serve its current customers better by providing announcements of new products, services, and deals. The potential is enormous and as varied as your imagination allows.

At this time, auto part stores that are on the Web only post company profile; customers wishing to purchase parts must submit a request form and wait for a response. I am proposing to build an E-commerce site that will allow secure online purchasing.

3. Description of the Solution

Small businesses that do not have a sales force or the revenue needed to market their products through traditional means can now do it using the power of the World Wide Web. With the emergence of e-commerce, more and more businesses are putting their stores on the Internet. I will develop a secure e-commerce site that will market as well as sell automotive parts for Cincinnati Generators and Starters.

3.1 User Profile

Users of this system are more than just the people who connect to the Web site to order parts. The system is based on user-centered design; any individual who has any contact with the system is a user of the system. Naturally, the end-user, accessing the system via the Web, is a user of the system. The person responsible for maintaining the parts inventory database is also a user of the system. The users in this case are not technically savvy. The site is geared toward garages, parts stores, and self-maintenance vehicle owners. The user maintaining the database has never worked with a database let alone maintaining one. The end-user may not be familiar with the system since this is a new approach to selling alternators and starters online.
The system must support the following requirements in order for the user to be able to effectively use the system:

- The system must be easy to use, as not all users will be technically savvy.
- The system must support searching for parts by vehicle year, make, and model. Not all users know the part number for their starter or alternator.
- The system must support credit card payment over the web or allow the shopper to give credit card information over the phone.
- The system must be easy to update so that new vehicles and parts can be added.

### 3.2 Design Protocol

The point of design this system is to sell products and to inform customers of available products. I started with a high-level task statement. “Purchase a part” and broke down the sub-tasks in it required to support e-commerce.

- Identify customer
- Search vehicle for parts carried
- Select quantity
- Add more selections to the order
- Make purchase

The five sub-tasks encompass all of the high level steps necessary to purchase an automotive part electronically. To support the task of searching for a part and then purchasing that part, a database was used to store information about the customer and the vehicle parts.

### 4. Deliverables

- Develop a SQL database for this project with a front-end data entry form allowing the client to easily add new vehicles and new parts to the database.
The site allows user interaction. Customers can go to another page and send email with a click of the mouse.

Users are able to sign on as customers and search the database for vehicle parts. The database contains user information so that the same data does not have to be entered twice.

Users are able to view and add items to the shopping cart before making a purchase.

User-centered design that is easy to use. Customers can easily navigate to all the pages.

5. Design and Development

5.1 Budget

From the start this project has been a test case for CGS and an opportunity for me to expand on the courses I have taken. The total budget was kept at a minimum.

**Break down of the budget:**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL server 7 trial version</td>
<td></td>
</tr>
<tr>
<td>MS Visual InterDev 6</td>
<td>$0</td>
</tr>
<tr>
<td>MS Visual Basics 6</td>
<td>$0</td>
</tr>
<tr>
<td>Macromedia Fireworks demo/cracked</td>
<td></td>
</tr>
<tr>
<td>Reference books</td>
<td>$100</td>
</tr>
</tbody>
</table>

I am using the server in the fourth floor lab and my home PC as development PCs.

Since I own a copy of MS Studio 6.0 and the trial version of SQL Server, there will be no budget for these items. I have budgeted $100 for reference books for this project. Because this is a test case for the client I will use the demo version of the software needed for the project.
5.3 Software

As stated earlier, this project is a test case and an opportunity for me to expand on the development tools I have learned in this major however I did take a look at other web development tools that were available at the time. Allaire ColdFusion 4.0, Visual Studio 6.0, and Macromedia Dreamweaver 2 were considered. ColdFusion 4.0 comes with Homesite as its Web page builder. This easy to use Web development tool uses special tags that are known to ColdFusion and requires the use of ColdFusion server. Dreamweaver was the other Web development tool taken into consideration for its maximum control over design capabilities however it lacked database capabilities that other packages offered. Visual Studio 6 is the only package that offered the complete development tool. Visual InterDev 6 is a fully functional RAD Web development environment that makes use of ASPs for delivering Web content on the fly. This development tool had a lower learning curve than all the other software packages I have considered due to the fact that I have had courses in Visual InterDev and Visual Basics.

5.4 Hardware

Hardware was not much of an issue because it is uncontrollable since I choose to use the fourth floor lab servers. The requirements were that the server had SQL Server installed, IIS installed, and a dedicated Internet connection.

Figure 2. IET Lab Servers
6. Proof of Design

6.1 Data Architecture

To support the task of searching for a starter or alternator electronically and then purchasing it, there needs to be a data architecture that corresponds to the task. Here are three main areas that need to be supported.

- The user/shopper
- Parts catalog
- An electronic shopping cart/order

Within these areas there are one or more supporting data tables that will contain the information necessary to support the user’s task.

6.1.1 User data

The user data is what is used to identify the particular user that is placing an order in the system. The customer is assigned an ID that is unique to them. The ID is used to identify the customer’s information within the system, so that information on the customer does not have to be re-entered. The Customer structure has the following structure.
6.1.2 Parts Data

The data requirements for the auto parts in the catalog are more extensive than the user data requirements. The database must support the information about the parts, which includes:

- Year, make and model of vehicle
- Price
- Product Name
- Picture file name
- Description

To support the parts information in the database, the parts table was used with the following structure.

You may notice that price and vehicle information were missing in the table. One starter may go on several vehicles and the price varies depending on what vehicle it goes on and what vehicle year it is for. Having multiple vehicles and prices within the parts
The table would solve this but would violate the normalization rules of a relational database.

To effectively support this requirement, I created two separate tables, Vehicles table and VehicleParts table, to link vehicle to parts. The Vehicles table has the following structure:

![Figure 5. Design View of the Vehicles Table](image)

The following is the VehicleParts table structure:

![Figure 6. Design View of the VehicleParts Table](image)

By linking vehicles to parts in a separate table, it allows CGS to price parts according to the vehicle year, make and model.

6.1.3 Order Data

The third major section of the data is the order information. The data actually provides two functions. As the customer proceeds through the system and adds selection to their electronic shopping cart, that information is tracked by the system. The shopping
cart data consists of the quantity of each part that the customer is buying. Once the customer checks out, the content of the shopping cart becomes the order information.

There are four tables that support the order data. The first is the Shopcart table. This table is used to hold information about the shopping basket. The information includes Internet session ID, part ID, quantity, and price. The information stored in this table is transferred to the orders table and deleted when the customer checks out. The table structure is as follows:

![Design View of tbl_shopcart Table](image)

**Figure 7. Design View of tbl_shopcart Table**

The second table is the session_ref table, which ties the customer to the shopping cart items. This table is used as a reference in getting the customer information from the customer table. The columns in this table are session_id and customer_id.

The third table is the orders table. This table is used to hold information about the purchase. This information includes who the customer is, the order number, order date, and credit card information. The table structure for this table is as follows:
The fourth table is the OrderDetail table. To store the content of the order, the OrderDetail table is used. This table contains the ID of the vehicle part, which was placed in the shopcart table. The structure of this table is:
6.2 Application Architecture

The core features of any e-commerce site are the catalog features. Customers must be able to browse the product catalog or search for products they want to purchase. The e-commerce site must have the following features:

- Product search
- Shopping cart
- Login and Registration
- Checkout

6.2.1 Implementing Product Search

In dealing with CGS I thought searching for parts for a particular vehicle is most sensible. When customers buy auto parts from the local store they usually look up the year, make, and model of their vehicle to find the parts they need for that specific vehicle. I designed the search so that it is fool proof. The smart search populates the fields from which the customer to choose. When the customer selects the year of his vehicle, select make field is populated with all the makes that were available for that year. Next the customer selects the make of his vehicle, the select model field is populated with all the models available for that year and that make. The customer then can select the model of his vehicle. (See appendix A for stored procedures.) The customer can check his selections below before submitting the search. The data fields are being populated directly from the database. When the database is updated the changes are automatically displayed, there is no additional code.
Figure 10. Part Search by Vehicle

The selected year, make and model is passed to the results page where it is entered as the search parameter of the stored procedure. (See Appendix A.) The results of the stored procedure are displayed in a data grid.
Figure 11. Search Results

There are two hot spots in the grid. Under WAI Number and Add to Cart columns are links that the customer can click. Clicking on the WAI number take the customer to the parts detail page, where the he can view the details about the part in question. In the parts detail page the customer will find an image of the part along with other information such as the particular vehicle that it applies to, amp in kW, and price. (See Appendix A.)
6.2.2 Implementing Shopping Cart

Clicking on the “Click to Buy” link will direct the customer to the shopping cart. The shopping cart displays all the items the customer adds to the basket during this session.

There are four parts that make up the shopping cart: add, remove, clear all, and shopcart.

Add, remove and clear all are action pages, therefore no data is will be physically displayed. These ASPs simply add, delete, or clear the items in the customer’s shopping cart and redirects the customer to the shopcart page where changes to the shopping cart are displayed. (See Appendix B, C, and D.)

Creating the shop cart page was much simpler than using DTC. A recordset DTC was used to retrieve the current session records from the tbl_shopcart. The results were
displayed in the grid. Additional functionality was added at the bottom of the page to present totals and allow some navigation.

Figure 13. Shopping Cart

6.2.3 Login and Registration

Figure 14. Login Page
The login and registration forms were designed using drag and drop controls and are straightforward. The registration form took some coding to check the fields before passing the entered data to newaccount.asp page. The newaccount.asp pages retrieves the data entered in the previous form and adds it to the SQL database. The added data are displayed to the customer showing him what was added.

![New Account Registration](image)

**Figure 15. New Account Registration**

There are two processes in the login as well. After the customer enters his email address and password it is submitted to db_addsession_ref.asp. This action page has two functions. It checks the e-mail address and password to see if it matches the database. If the entered e-mail address is not in the database it redirects the customer to newaccounts.htm. If the email address were correct the customer is redirected to login.htm. When the entered data matches a record in the table, the e-mail address and the current session ID is entered in the session_ref table.
6.2.4 Checkout

Checkout is made up of two parts, checkout1.asp and checkout2.asp. Checkout1 displays the customers shipping and billing information it retrieved from the customer table and the shopping cart items. The customer must enter credit card information before proceeding to checkout2.asp. (See Appendix E.)
Checkout2.asp does three functions:

- It checks the credit card to verify that the number given is an actual credit card number using an algorithm.
- It enters the shopping cart items into the orders table.
- It displays a printable receipt of the order.
6.3 Web Layout

There are two layouts used throughout the Web site. The layouts are shown below.

![Basic Layouts of the Site](image)

**Figure 19. Basic Layouts of the Site**

The layout on the left is the home page layout and the one on the right is the child layout. From everywhere in the site the customer is able to navigate to all pages of the Web site using the links on the left.
6.4 Data Entry Application

Figure 20. Data Entry Application

Figure 20.1 Vehicles and Parts Data Entry Form
Data entry forms were developed using Visual Basics 6. It provides CGS a simple solution to entering data. Through use of ADO, I developed this simple application in a day. Most of that time was spent on creating visual effects. The actual code is no more than 20 lines for each form. As seen in figure 20.3, the user can associate as many vehicles to this part as he wishes. The WAI Num field of the Part form is automatically entered to WAI Num of the Vehicle Part form. This helps eliminate user error. All the user has to do is enter the vehicle ID and the price associated to that vehicle part. (See Appendix F.)
7. Conclusion and Recommendations.

I recommend the CGS to test the market for online ordering of auto parts. Adding a new source for selling and marketing product carried in CGS can only help the company. CGS should first, setup the online catalog to measure the need for online ordering of generators and starters. It makes good business sense to have a Web site even if it serves as an online catalog or marketing presence. E-commerce allows store to market and sell nationally.
Appendix A

Stored Procedure

CREATE PROCEDURE addOrder
    @cust_id int,
    @ccnum nvarchar(20),
    @cardexp varchar(10),
    @cardtype varchar(15),
    @subtotal money,
    @shiprate money,
    @tax money,
    @total money,
    @sessionID nvarchar (10),
    @orderdate  datetime
as
begin tran

    insert into Orders
    (Customer_ID, OrderDate, cc_type, cc_number, cc_exp, total_product, total_ship, total_taxes, total_grand)
    Values(@cust_id, @orderdate, @cardtype, @ccnum, @cardexp, @subtotal, @shiprate, @tax, @total)
    Select @@identity

    insert into OrderDetail
    (Order_ID, PKPart, price, qty)
    select Orders.Order_ID, tbl_shopcart.prod_id, tbl_shopcart.price, tbl_shopcart.qty
    from Orders, tbl_shopcart
    where Orders.Order_ID=@@Identity and tbl_shopcart.session_id = @sessionID

    delete from tbl_shopcart where session_id=sessionID
    delete from session_ref where session_id=sessionID

    commit tran

---

CREATE PROCEDURE addSession_ref
    @email char(30),
    @session char(10)
AS
begin tran
insert into session_ref
values(@email, @session)
commit tran

select * from session_ref where session_id = @session

CREATE PROCEDURE checkout
@session nvarchar (10)
as
SELECT Customer.Company_Name, Customer.FName,
   Customer.LName, Customer.Street, Customer.City,
   Customer.State, Customer.ZipCode,
   Customer.Shipping_Address, tbl_shopcart.prod_id,
   tbl_shopcart.prod_name, tbl_shopcart.price,
   tbl_shopcart.qty
FROM Customer INNER JOIN
   session_ref ON
   Customer.Email_Address = session_ref.Email_Address INNER JOIN
   tbl_shopcart ON
   session_ref.session_id = tbl_shopcart.session_id
where session_ref.session_id = @session

CREATE PROCEDURE QueryPart
@year char(4),
@make char(12),
@model char(25)
AS
SELECT Vehicles.YEAR, Vehicles.MAKE, Vehicles.MODEL,
   Vehicles.Vehicle_ID,
   Vehicles.ENG_CYL, Vehicles.ENG_LITER, VehicleParts.Price,
   VehicleParts.PKPart,
   Parts.PROD_NAME, Parts.AMP_KW, Parts.WAI_NUM
FROM Vehicles INNER JOIN
   VehicleParts ON
   Vehicles.Vehicle_ID = VehicleParts.Vehicle_ID INNER JOIN
   Parts ON VehicleParts.WAI_NUM = Parts.WAI_NUM
WHERE (Vehicles.YEAR =@year) AND (Vehicles.MAKE =@make) AND
   (Vehicles.MODEL = @model)
Appendix B

Add Item to Shopcart

```vbnet
%@Language=VBScript %>
<SCRIPT id=DebugDirectives runat=server language=javascript>
// Set these to true to enable debugging or tracing
@set @debug=false
@set @trace=false
</SCRIPT>

%@ Response.Buffer = true
'on error resume next %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<% function prod_exists(byVal prod_id, byRef qty, byRef oconn)
    sqltext="Select prod_id, qty from tbl_shopcart " &
        "WHERE session_id = " & session.sessionid & " " &
        "AND prod_id = " & prod_id
    set rs=oconn.Execute(sqltext,affect)
    if( rs.eof or rs.bof) then
        err.clear
        prod_exists = false
        exit function
    else
        qty = qty + rs("qty")
        err.clear
        prod_exists = true
        exit function
    end if
end function
%

<% function add_item(byVal session_id, byVal prod_id, byVal qty, byVal oconn)
    sqltext="Insert into tbl_shopcart (session_id, prod_id, qty) VALUES " &
        "(" & session.sessionid & ", " & prod_id & ", " & qty & ")"
    set rs=oconn.Execute(sqltext,affect)
    if( rs.eof or rs.bof) then
        err.clear
        prod_exists = false
        exit function
    else
        qty = qty + rs("qty")
        err.clear
        prod_exists = true
        exit function
    end if
end function

'****************************************************************************
' function: add_item
' parameters: session_id, prod_id, qty, oconn
' purpose:
'    add the passed item to the shopping cart
'****************************************************************************
```
function add_item(byVal session_id, byVal prod_id, byVal qty, byRef oconn)

    ' Get the current name and price from the database
    sqltext = "Select PROD_NAME, Price from ProductDetail " & _
        "WHERE PKPart = " & prod_id

    set rs = oconn.Execute(sqltext)
        prod_name = rs("PROD_NAME")
        price = rs("Price")

    ' Update the shopping cart with current data
    sqltext = "INSERT INTO tbl_shopcart VALUES(" & session_id & "," & prod_id & "," & prod_name & "," & Price & "," & qty & ")"
    oconn.Execute (sqltext)
    '*** <<<check>>> Response.Write sqltext

    if Err.number <> 0 then
        '** Handle Error **'
        Err.Clear
    end if
end function

<%>

'*****************************************************
' purpose:
'    update the count of an item if it already exists
' in the shopping cart
'*****************************************************

function update_count(byVal prod_id, byVal qty, byRef oconn)
    sqltext = "UPDATE tbl_shopcart SET qty = " & qty & "," & _
        "WHERE( session_id = " & session.SessionID & "," & _
        "AND prod_id = " & prod_id & ");"

    oconn.Execute sqltext

    if Err.number <> 0 then
        '** Handle Error **'
        Err.Clear
    end if
end function

<%>

straint

***************
' purpose:
'    update the count of an item if it already exists
' in the shopping cart
'******************************************
dim qty, prod_id, session_id, oconn
    qty = Request.QueryString("qty")
    prod_id = Request.QueryString("PKPart")
    session_id = session.SessionID

set oConn = Server.CreateObject("ADODB.Connection")

    oConn.Open "VictorDB","hsu","password"

if( prod_exists(prod_id, qty, oConn) ) then
    update_count prod_id, qty, oConn
else
    add_item session_id, prod_id, qty, oconn
end if

oConn.Close
set oConn = nothing

if Err.number <> 0 then
    '** Handle Error **'
    Err.Clear
else
    Response.Redirect ("shopcart.asp")
end if

%</BODY>

<P>&nbsp;</P>

</HTML>
Appendix C

Delete Item from Shopcart

```vbscript
<%@ Language=VBScript %>
<% Response.Buffer = True
on error resume next %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<%
'*****************************************************************************
' function: remove_item
' parameters: prod_id, oconn
' purpose: remove an item from the shopcart
'*****************************************************************************
sub remove_item(byVal prod_id, byRef oconn)
    sqltext = "DELETE from tbl_shopcart " &
             "WHERE session_id = " & session.SessionID & " " &
             "AND prod_id = " & prod_id
    oconn.Execute sqltext
    if Err.number <> 0 then
        '** Handle Error **'
        Err.Clear
    end if
end sub
%
<%
'*****************************************************************************
set oconn = Server.CreateObject("ADODB.Connection")
oconn.Open "VictorDB","hsu","password"

call remove_item(Request.QueryString("prod_id"), oconn)
oConn.Close
set oconn = nothing

Response.Redirect ("shopcart.asp")
%
</BODY>
</HTML>
```
Appendix D

Clear All Item from Shopcart

<%@ Language=VBScript %>
<% Response.Buffer = true
    on error resume next %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<%
'*********************************************************************
' function: remove_all
' parameters: oconn
' purpose: remove all items from shopcart
'*********************************************************************
sub remove_all(byRef oconn)
    sqlText = "DELETE from tbl_shopcart " &
                 "WHERE session_id = " & session.SessionID

    oconn.Execute sqltext

    if Err.number <> 0 then
        '** Handle Error **'
        Err.Clear
    end if
end sub
%

<%
set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "VictorDB","hsu","password"
call remove_all(oconn)
oConn.Close
set oConn = nothing

Response.Redirect ("shopcart.asp")
%

</BODY>
</HTML>
Appendix E

Checkout2.asp

```vbscript
<%@ Language=VBScript %>
<script id="DebugDirectives" runat="server" language="javascript">
// Set these to true to enable debugging or tracing
@set @debug=false
@set @trace=false
</script>
<% ' VI 6.0 Scripting Object Model Enabled %>
<!--#include file="_ScriptLibrary/pm.asp"-->  
<% if StartPageProcessing() Then Response.End() %>  
<form name="thisForm" METHOD="post">
<html>
<head>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<!-- #include file="adovbs.inc" -->
</head>
<%
    dim ccnum, cardexp, cardtype, cardholder, newidentity
    dim total_product, total_ship, total_tax, total_grand, text
    dim company, fname, lname, address, city, state, zip
    dim shipping, cardnum, stryear, strmake, strmodel, engine, product
    dim ship_address, ship_citystatezip, icomma, length, searchr, lenRight, lenLeft
    dim amp, description, msg, vlink

    cardholder = Request.Form ("cardholder")
    cardtype = Request.Form ("cardtype")
    ccnum = Request.Form ("cardnumber")
    cardexp = Request.Form ("cardexp")
    total_product = Request.Form ("txtSubtotal")
    total_ship = Request.Form ("txtShip")
    total_tax = Request.Form ("txtTax")
    total_grand = Request.Form ("txtTotal")
    text = Request.Form ("text1")
    'Response.Write "Hope this works:" & total_grand & total_tax & total_product
    'Response.Write date()
    function calc_price(price, qty)
        dim cartprice
        cartprice = cartprice + (price * qty)
        cale_price = formatcurrency((price * qty))
    end function

    function format_price(price)
```
format_price = formatcurrency(price)
end function

<% 

'Create two variables with global scope
Dim paymethod
Dim TheCardType

TheCardType = cardtype
'msgbox "does this work?" & cardexp
call VerifyCard()
'Response.Write "you've selected:" & TheCardtype
call verifyCard
'This is the main sub routine and is called by the button's onclick event handler
Sub VerifyCard
    dim strCleanNo
    dim verified
dim currdate

'check the date if bad then quit
If Not GoodDate() Then
    Exit Sub
End If

'take all spaces etc out of the string
strCleanNo = CleanString()

'determine the type of card and send execution off in the right direction
Select Case TheCardType
    Case "amex"
        verified = verifyAmex(strCleanNo)
    Case "visa"
        'this is a nested select
        Select Case Len(strCleanNo)
            Case 13
                verified = verifyV(strCleanNo)
            Case 16
                verified = verifyMC(strCleanNo)
            Case Else
                verified = False
        End Select
    Case "mastercard"
        verified = verifyMC(strCleanNo)
    Case "discovery"
        verified = verifyMC(strCleanNo)
    Case Else
        msg = "Please select a card type" &"<BR>"
        vlink = "<a href=checkout1.asp>Return to Previous Page</a>"
        Exit Sub
End Select

'is it ok or not?
If Not verified Then
    msg = "INVALID CARD NUMBER. Cannot Submit Data <BR>"
    vlink = "<a href=checkout1.asp>Return to Previous Page</a>"
    Exit Sub
Else
    msg = "The Card No. is OK. Submitting Data....<BR>"
    msg = msg + "Please print this page to as your receipt"
End If
call AddtoOrder()
rs_invoice.open
call getfields()
End If
End Sub

'This is the function that checks the date
'
Function GoodDate()
GoodDate = True
If Not IsDate(cardexp) Then
    msg = "Invalid Expiry Date. Cannot Submit Data" &"<BR>"
    vlink = "<a href=checkout1.asp>Return to Previous Page</a>"
    GoodDate = False
Else
    currdate = Month(Now()) & " " & Year(Now())
    If DateValue(cardexp) < DateValue(currdate) Then
        Response.Write "THIS CARD HAS EXPIRED. Cannot Submit Data"
        Response.Write "<a href=checkout1.asp>Return to Previous Page</a>"
        GoodDate = False
    End If
End If
End Function

'This function validates an amex card
'
Function VerifyAmex(ByVal CardNo)
    Dim amexarray(14)
    Dim tot
    tot=0

    If Len(CardNo) <> 15 Then
        VerifyAmex = False
        Exit Function
    End If

    For x = 2 to 10 step 2
        amexarray(x-1) = Mid(CardNo,x,1)
    Next

    For x = 12 to 15
        amexarray(x-1) = Mid(CardNo,x,1)

    End For
End Function
For x = 1 to 11 Step 2
    y = Mid(CardNo,x,1) * 2
    If y >= 10 Then
        amexarray(x-1) = (y Mod 10) + 1
    Else
        amexarray(x-1) = y
    End If
Next

For x = 0 to 14
    tot = tot + CInt(amexarray(x))
Next

If tot Mod 10 = 0 Then
    VerifyAmex = True
Else
    VerifyAmex = False
End If
End Function

' This function checks visa 13 digit cards
Function VerifyV(ByVal CardNo)
    Dim vArray(12)
    Dim tot
tot=0

    For x = 2 to 12 step 2
        vArray(x-1) = Mid(CardNo,x,1)
    Next
    vArray(12) = Mid(CardNo,13,1)

    ' Mod returns the remainder of a division
    For x = 1 to 11 Step 2
        y = Mid(CardNo,x,1) * 2
        If y >= 10 Then
            vArray(x-1) = (y Mod 10) + 1
        Else
            vArray(x-1) = y
        End If
    Next
For x = 0 to 12
    tot = tot + CInt(vArray(x))
Next

If tot Mod 10 = 0 Then
    VerifyV = True
Else
    VerifyV = False
End If

End Function

'Although it says MC it's actually a multi purpose 16 digit checker

Function VerifyMC(ByVal CardNo)
    Dim tot
    tot=0

    If Len(CardNo) <> 16 Then
        VerifyMC = False
        Exit Function
    End If

    For x = 2 to 16 step 2
        tot = tot + CInt(Mid(CardNo,x,1))
    Next

    For x = 1 to 15 Step 2
        y = Mid(CardNo,x,1) * 2
        If y >= 10 Then
            tot = tot + CInt((y Mod 10) + 1)
        Else
            tot = tot + CInt(y)
        End If
    Next

    If tot Mod 10 = 0 Then
        VerifyMC = True
    Else
        VerifyMC = False
    End If

End Function
'This function looks for numbers in the card number string, thereby removing all non numeric characters

Function CleanString()
    dim strLen
    dim strCounter
    dim strClean
    dim strDirty

    strDirty = Trim(ccnum)
    strLen = Len(strDirty)
    strClean = ""

    For strCounter = 1 to strLen
        If asc(Mid(strDirty,strCounter,1)) < 58 AND Asc(Mid(strDirty,strCounter,1)) > 47 Then
            strClean = strClean & Mid(strDirty,strCounter,1)
        End If
    Next

    CleanString = strClean
End Function

sub AddtoOrder()

    set objcmd = server.CreateObject ("ADODB.Command")

    with objcmd
        .ActiveConnection = oconn.ConnectionString
        .CommandType = 4
        .CommandText = "addOrder"
        .Parameters ("@cust_id")= cust_id
        .Parameters ("@ccnum")= ccnum
        .Parameters ("@cardexp")= cardexp
        .Parameters ("@cardtype")= cardtype
        .Parameters ("@subtotal")= total_product
        .Parameters ("@shiprate")= total_ship
        .Parameters ("@tax")= total_tax
        .Parameters ("@total")= total_grand
        .Parameters ("@sessionID")= session_id
        .Parameters ("@orderdate")= today
        .Parameters ("")=
    end with

    set rs_addorder = objcmd.Execute(iRecs)
newidentity=rs_addorder(0)
Response.Write newidentity
end sub
%
</%
Sub getfields()

with rs_invoice.fields
    company= .getValue ("Company_Name")
    fname= .getValue ("FName")
    lname= .getValue ("LName")
    address= .getValue ("Street")
    city= .getValue ("City")
    state= .getValue ("State")
    zip= .getValue ("ZipCode")
    shipping= .getValue ("Shipping_Address")
    cardnum= .getValue ("cc_number")
    stryear= .getValue ("Year")
    strmake= .getValue ("Make")
    strmodel= .getValue ("Model")
    engine= .getValue ("ENG_CYL")
    product= .getValue ("PROD_NAME")
    amp= .getValue ("AMP_KW")
end with

' the following code gets the street address and city, state, zip code out of a text string

'shipping=Recordset1.fields.getValue ("Shipping_Address")
searchr= ";"
'***get length of string***
length = len(shipping)
'***get position for "",
lenLeft=instr(1,shipping,searchr)
'*** calculate length of string for city, state, zip
lenRight=length - lenLeft
'***string manipulation to get street address and city, state and zip code
ship_address = Left(shipping, lenLeft - 1 )
ship_citystatezip = right(shipping,lenRight -1)

description= product & " for " & stryear & " " & strmake & " " & strmodel & " with " & engine & " Cylinder Engine " & "(" & amp & " amp" & ")"
end sub
%>
function _setParametersrs_invoice()
{
    rs_invoice.setParameter(0,newidentity);
}
function _initrs_invoice()
{
    rs_invoice.advise(RS_ONBEFOREOPEN, _setParametersrs_invoice);
    var DBConn = Server.CreateObject('ADODB.Connection');
    DBConn.ConnectionTimeout = Application('ConnSQL_ConnectionTimeout');
    DBConn.CommandTimeout = Application('ConnSQL_CommandTimeout');
    DBConn.CursorLocation = Application('ConnSQL_CursorLocation');
    DBConn.Open(Application('ConnSQL_ConnectionString'),
                Application('ConnSQL_RuntimeUserName'),
                Application('ConnSQL_RuntimePassword'));
    var cmdTmp = Server.CreateObject('ADODB.Command');
    var rsTmp = Server.CreateObject('ADODB.Recordset');
    cmdTmp.ActiveConnection = DBConn;
    rsTmp.Source = cmdTmp;
    cmdTmp.CommandType = 1;
    cmdTmp.CommandText = 'SELECT Customer.*, Orders.OrderDate ,
                        Orders.Ship_via, Orders.Order_ID, Orders.ShipDate, Orders.cc_number,
                        Orders.total_product, Orders.total_ship, Orders.total_taxes, Orders.total_grand,
                        OrderDetail.PKPart, OrderDetail.qty, OrderDetail.Price, Parts.PROD_NAME,
                        Parts.AMP_KW, Vehicles.YEAR, Vehicles.MAKE, Vehicles.MODEL,
                        Vehicles.ENG_CYL FROM Customer INNER JOIN Orders ON
                        Customer.Customer_ID = Orders.Customer_ID INNER JOIN OrderDetail ON
                        Orders.Order_ID = OrderDetail.Order_ID INNER JOIN VehicleParts ON
                        OrderDetail.PKPart = VehicleParts.PKPart INNER JOIN Parts ON
                        VehicleParts.WAI_NUM = Parts.WAI_NUM INNER JOIN Vehicles ON
    rsTmp.CacheSize = 10;
    rsTmp.CursorType = 3;
    rsTmp.CursorLocation = 3;
    rsTmp.LockType = 3;
    rs_invoice.setRecordSource(rsTmp);
}
function _rs_invoice_ctor()
{
    CreateRecordset('rs_invoice', _initrs_invoice, null);
}
</SCRIPT>

<!--METADATA TYPE="DesignerControl" endspan-->
<table border="0" width="645" style="HEIGHT: 482px; WIDTH: 645px"
<tr><td rowSpan="2" width="165" vAlign="top"><IMG height=86 src="images/cgs01.jpg" width=160></td></tr>
<table border="0" cellPadding="1" cellSpacing="1" style="HEIGHT: 312px; WIDTH: 154px" width="90%">
<tr><td style="VERTICAL-ALIGN: top">
<p><font size="2"><a href="Default.htm">Home</a></font></p>
<p><font size="2"><a href="CompanyProfile.htm">Company Profile</a></font></p>
<p><font size="2"><a href="Search.asp">Find Parts</a></font></p>
<p><font size="2"><a href="NewAccount.htm">Account Registration</a></font></p>
<p><font size="2"><a href="AlternatorTip.htm">Alternator Tips</a></font></p>
<p><font size="2"><a href="StarterTip.htm">Starter Tips</a></font></p></td></tr></table></td></tr>
<tr><td height="80"><p align="center"><IMG height=55 src="images/thankyou.jpg" width=303></p></td></tr>
<tr><td><p><%=msg%><br><%=vlink%></p>
<table border="0" cellPadding="1" cellSpacing="1" width="100%" style="WIDTH: 100%">
<tr><td><strong>Order #: <%=newidentity%></strong></td><td><strong>Order Date: <%=date()%></strong></td></tr>
<tr><td><strong>Bill To: </strong> <%=company%><br><%=fname & " " & lname%><br><%=address%><br><%=city &", " & state & " " & zip%></td><td><strong>Ship To: </strong> <%=company%><br><%=fname & " " & lname%><br><%=ship_address%><br><%=ship_citystatezip%></td></tr>
</table></td></tr>
<PARAM NAME="DetailFontColor" VALUE=",,,,">
<PARAM NAME="DetailFontSize" VALUE=",,,,">
<PARAM NAME="DetailFontStyle" VALUE=",,,,">
<PARAM NAME="ColumnCount" VALUE="5">
<PARAM NAME="CurStyle" VALUE="Simple List">
<PARAM NAME="TitleFont" VALUE="Arial">
<PARAM NAME="titleFontSize" VALUE="2">
<PARAM NAME="TitleFontColor" VALUE="0">
<PARAM NAME="TitleBackColor" VALUE="16777168">
<PARAM NAME="TitleFontStyle" VALUE="1">
<PARAM NAME="TitleAlignment" VALUE="2">
<PARAM NAME="RowFont" VALUE="Arial">
<PARAM NAME="RowFontColor" VALUE="0">
<PARAM NAME="RowFontStyle" VALUE="0">
<PARAM NAME="RowFontSize" VALUE="2">
<PARAM NAME="RowBackColor" VALUE="16777215">
<PARAM NAME="RowAlignment" VALUE="0">
<PARAM NAME="HighlightColor3D" VALUE="268435455">
<PARAM NAME="ShadowColor3D" VALUE="268435455">
<PARAM NAME="PageSize" VALUE="20">
<PARAM NAME="MoveFirstCaption" VALUE="    |<    ">
<PARAM NAME="MoveLastCaption" VALUE="    >|    ">
<PARAM NAME="MovePrevCaption" VALUE="    <<    ">
<PARAM NAME="MoveNextCaption" VALUE="    >>    ">
<PARAM NAME="BorderSize" VALUE="1">
<PARAM NAME="BorderColor" VALUE="8421504">
<PARAM NAME="GridBackColor" VALUE="16777215">
<PARAM NAME="AltRowBckgnd" VALUE="268435455">
<PARAM NAME="CellSpacing" VALUE="1">
<PARAM NAME="WidthSelectionMode" VALUE="1">
<PARAM NAME="GridWidth" VALUE="464">
<PARAM NAME="EnablePaging" VALUE="0"></OBJECT>

```javascript
function _initGrid1()
{
    Grid1.pageSize = 0;
    Grid1.setDataSource(rs_invoice);
    Grid1.tableAttributes = ' cellpadding=2 cellspacing=1 bordercolor=Gray bgcolor=White border=1 cols=5 rules=COLS WIDTH=464';
    Grid1.headerAttributes = ' bgcolor=#ffffd0 align=Center';
    Grid1.headerWidth[0] = ' WIDTH=55';
    Grid1.headerWidth[1] = ' WIDTH=264';
```
<p align="right"><strong>Shipping:</strong></p>
<td><input id="txtShip" name="txtShip" readOnly size="10" style="WIDTH: 60px" value="<%=total_ship%>" width="60"></td></tr>
<tr>
<td align="right"><strong>Tax: <br></strong><font size="2">(Applies to OH Residents Only)</font></td>
<td><input id="txtTax" name="txtTax" readOnly size="10" style="WIDTH: 60px" value="<%=total_tax%>" width="60"></td></tr>
<tr>
<td><p align="right"><strong>Total:</strong></p></td>
<td><input id="txtTotal" name="txtTotal" readOnly size="10" style="WIDTH: 60px" value="<%=total_grand%>" width="60"></td></tr></table>

</body>
<% ' VI 6.0 Scripting Object Model Enabled %>
<% EndPageProcessing() %>
</form>
</html>
Appendix F
Data Entry Application

**MDIForm1 – Source Code:**

```vbnet
Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
    Select Case Button.Key
    Case "Vehicle"
        frmVehicle.Show
    Case "Part"
        frmParts.Show
    Case "Exit"
        Unload frmVehicle
        Unload frmParts
        Unload frmVehPart
        Unload Me
    End Select
End Sub
```

**frmVehicle – Source Code:**

```vbnet
Private Sub cmdClose_Click()
    Unload frmVehicle
End Sub

Private Sub cmdNewRec_Click()
    Dim strcar
    strcar = txtYear & " " & txtMake & " " & txtModel
    Adodc1.Recordset.Update
    Debug.Print strcar
    MsgBox strcar & Chr$(10) & "Added to Database", vbOKOnly, "New Record"
    Adodc1.Recordset.MoveNext
End Sub

Private Sub Form_Load()
    Adodc1.Recordset.AddNew
End Sub
```

**FrmParts – Source code:**

```vbnet
Private Sub Adodc1_MoveComplete(ByVal adReason As DODB.EventReasonEnum, _
    ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, _
    ByVal pRecordset As ADODB.Recordset)
    PartType
End Sub
```
Private Sub cmdVehPart_Click()
    frmVehPart.Show
End Sub

Private Sub cmdClear_Click()
    Adodc1.Recordset.AddNew
    PartType
End Sub

Private Sub Form_Load()
    Adodc1.Recordset.AddNew
    PartType
End Sub

Sub PartType()
    If Adodc1.Recordset.Fields("PROD_NAME") = "ALTERNATOR" Or
    Or Adodc1.Recordset.Fields("PROD_NAME") = "Alternator" Or
    Or Adodc1.Recordset.Fields("PROD_NAME") = "alternator" Then
        Option1(0) = True
    ElseIf Adodc1.Recordset.Fields("PROD_NAME") = "" Then
        Option1(0) = False
        Option1(1) = False
    Else
        Option1(1) = True
    End If
End Sub

Private Sub cmdAddPart_Click()
    Dim prod_name
    For i = 0 To 1
        If Option1(i) = True Then
            prod_name = Option1(i).Caption
        End If
    Next i
    Debug.Print prod_name
    If prod_name = "" Then
        MsgBox "You must select either Alternator or Starter to Continue", vbOKOnly,
        "Required Field Empty"
    Else
        Adodc1.Recordset.Fields("PROD_NAME") = prod_name
        Adodc1.Recordset.Update
        strWai = txtWai.Text
        MsgBox "New " & prod_name & " was added", vbOKOnly, "New Part"
cmdVehPart.Enabled = True
cmdClear.Enabled = True
End If
End Sub

frmVehiclePart – Source code:
Private Sub cmdAddVehpart_Click()
    Adodc1.Recordset.Update
    Adodc1.Recordset.AddNew
End Sub

Private Sub cmdClose_Click()
    Unload frmVehPart
End Sub

Private Sub Form_Load()
    Adodc1.Recordset.AddNew
    txtWaiLink.Text = strWai
End Sub

FrmSplash – Source Code:
Private Sub Form_Load()
    MDIForm1.Show
End Sub

Private Sub Timer1_Timer()
    Unload Me
End Sub

Module1 – Code:
Public strWai As String
References


5.2 Timeline

Figure 1. Timeline