Spring 4-1-2005

Customer & Transaction Database with Custom Interfaces

Ross Denholm
Dakota State University

Customer & Transaction Database with Custom Interfaces

A Project Report submitted to the

Graduate Faculty

By

Ross Denholm

In partial fulfillment of the requirements for the degree of

Master of Science in Information Systems

Dakota State University

April 2005

DAKOTA STATE UNIVERSITY

MSIS
PROJECT APPROVAL FORM

Student Name: Ross Denholm

Expected Graduation Date: 5-7-05

Master's Project Title: Customer & Transaction Data base with Custom Interfaces

Date Project Plan Approved: Fall 04

Date Project Coordinator Notified and Grade Submitted: 5-4-05

Approvals/Signatures:

Student: _____ Date: 5-4-05

Faculty supervisor: _____ Date: 5/4/05

Committee member: _____ Date: 5/4/05

Committee member: Mark Moran Date: 5/4/05

Revised 1/28/03

Abstract

This document outlines the creation of a customer and transaction information system. The database includes both customer and transactional information, primarily to facilitate access to customer data and provide summary contract information. This database interacts with two different front-ends to simplify its usage and to provide additional functionality. One front end is created in Visual Basic.NET and allows a user to add, view, or edit any transaction or customer information. This front end allows a user to query the database for any desired information. The other front end is an ASP.NET web interface which provides similar functionality in an internet-based application.

Table of Contents

List of Figures

I. Introduction

Lightspeed Technologies is a small computer repair business in Madison, SD. In addition to typical PC repair, the shop sells computer and networking hardware and provides network support services. This full-service nature has led to the establishment of several service contracts, but there is currently no method in place to perform any kind of analysis on the contracts. In addition, the growing customer base has brought the need for some kind of customer information database to the forefront.

There is effectively no existing electronic system aside from a small amount of information haphazardly gathered through e-mail communication: a small address book consisting of contact information for a few customers. Since the company is owned and operated by a single person, there has not been time to implement any kind of database. Most information is currently recorded on paper records only. Very little customer-specific information is recorded independently, only appearing on the transactional receipts that are manually created for each transaction.

The purpose of this project is not to replace the current paper system because it would be impractical to do so due to the on-site nature of most of the business's transactions. Rather, this project is to serve as a supplementary information layer on top of the existing records. Due to this supplementary nature, much of the database design is derived from the current paper records, minimizing the redesign of any existing business processes.

II. Organizational Information

Organization Objectives:

The objective of Lightspeed Technologies is to provide high-quality, inexpensive computer repair and support services for both businesses and individuals.

Organizational Structure:

Lightspeed Technologies is a small company owned and operated by a single individual. Accounting services are provided by an outside firm, as are legal services, but almost all business activities are handled by the business owner.

While the structure is currently limited to a single individual, it is quite likely that an employee will be added in the near future, making intra-organizational communication an issue.

Description of Operations:

Computer Repair – Lightspeed's primary function is full-service computer repair for individuals and businesses. There is a strong emphasis on quality customer service in order to encourage customer loyalty in a competitive local market. Repair services include both hardware and software repair.

Networking Support – A natural extension of computer repair, Lightspeed also provides networking support services for both individuals and businesses. The supported networks vary from small, two computer peer-to-peer networks to 30 computer client-server business networks spanning two cities. The installation and maintenance of networking hardware is the primary function, although server configuration and maintenance is also performed.

Customer Service – Customer service is an important competitive advantage for Lightspeed Technologies. All services are performed with the express intention of keeping the customer happy. For example, in the event that a problem is not completely corrected during the first service call, the problem will be corrected with another visit free of charge. Also, in the rare cases that Lightspeed is unable to repair a computer, the customer is not charged. These and other customer service initiatives create high customer loyalty and differentiate Lightspeed from other local competitors.

Business Rules:

The business rules for Lightspeed Technologies are not very extensive. Computer service is a largely unregulated industry and very few business rules have been created by the owner. Most of the business rules that do exist are largely beyond the scope of the project, mostly involving legal obligations requiring the report of some computer crimes. The relevant business rules mostly revolve around the way business processes are executed and aren't generally codified as rules. The most important business rule in the development of the database is the one-to-many relationship between customers and transactions; each customer can have multiple transactions. The only other major business rule with an impact on the project is the fact that partial payments are not accepted, although occasionally deposits are requested and allowed.

III. Problem Identification

Interview:

The first step in the planning of this project was an informal interview with the owner of Lightspeed Technologies. This interview provided the initial motivation for this project as well as most of the basic specifications. The interview initially showed only a strong desire for a database system to store customer information. There was specific mention of a need to look up customers by phone number in certain situations, so the database and interfaces should be able to support this search.

As the interview progressed, the owner also expressed a desire to track the transactions involved with Lightspeed's contractual customers as well as a desire to obtain summary information about those transactions. In addition, the owner expressed unfamiliarity with the operation of database software despite owning a copy of Microsoft Access. The owner's current knowledge of Access includes data entry, record deletion, and basic navigation skills but no knowledge of queries, reports, or any other more advanced features.

This lack of familiarity with database usage suggested the use of a front-end to provide relative ease-of-use for the database system. The initial desire was for a Visual Basic application with a relatively standard Windows interface. The project expanded to also include an ASP.NET web application which could allow for data access from remote locations in the future, although because of the limited computer resources within the organization this web application will not be in use at project completion.

The basic structure of the database was also discovered during the interview process, derived from the existing paper records. Since the information system being

developed will not function as a replacement for the current paper system, it was logical to derive the database design from the paper documentation in order to facilitate data entry while avoiding the re-engineering of existing business processes.

IV. System Request

The owner requested that the database and front-ends for that database be developed in order to provide searchable customer and transaction information to support business operations.

The specific requirements are as follows:

- The system should provide customer information.
- The system should provide transaction information.
- The system should provide an interface to allow easy querying of the database.
- The system should provide summary information about transactions to aid in the analysis of contracts.

V. The Development Process

The design of the Lightspeed database was almost completely derived from existing paper documents, so very little design effort was required. The only fields in the database which are not derived from the paper documents are the primary and foreign keys for the two tables.

The choice of the DBMS was more difficult, but the ultimate choice was Access. Limited computer and monetary resources at Lightspeed made it impractical to implement many DBMS options, including Oracle and SQL Server. MySQL was considered as a low-cost alternative to Oracle or SQL Server, but the owner of Lightspeed was uncomfortable with the idea of running the MySQL server process on the one available computer. In the end, Access proved to be the best choice for this project.

The front-ends generally followed a prototyping methodology. Partially functional 'betas' were created and tested before proceeding with development. This methodology was particularly effective because of the developer's unfamiliarity with the programming tools used. The prototypes helped the learning process by providing quick feedback on the effectiveness of the development.

Prototyping also made the testing process easier, allowing for incremental increases in program usability and stability. Also, since testing was performed on individual functional modules as development progressed, prototyping reduced the need for a large-scale test at the end of the implementation process.

Further maintenance, support, and development of this system will be handled by the project developer and the owner of Lightspeed as the need arises. Possible future development includes the deployment of the ASP.NET interface.

VI. Objectives and Deliverables

The goal of this project was to provide a database that stored customer and transaction information which could easily be queried. Lightspeed needed a relatively easy-to-use system, although since the owner is the only user the interface could be specifically tailored to his preferences. In order to provide this ease-of-use, front-ends would be created in VB.NET and ASP.NET. Both front-ends needed to provide the ability to add, edit, and view database entries. Further, both front-ends needed to provide the ability to search for records in the database as well as specific summary information used to analyze contracts.

The deliverables include: the VB.NET executable, the ASP.NET scripts, and a sample database for demonstration purposes. Much of the VB.NET code is in Appendix A, although code generated by Visual Studio has been omitted to avoid excessive length. All ASP.NET code is included in Appendix B.

There were no significant changes to the original project plan. Aside from normal deviations in schedule and relatively minor alterations to the original VB.NET design, nothing from the planning stages was changed.

VII. Database Design

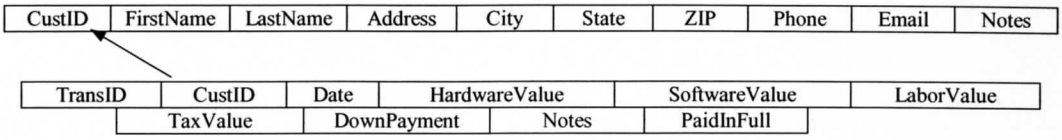


Figure 7.1: Relational Schema

The database was primarily designed based on the paper records already in use at Lightspeed. This was done to facilitate data entry into the database as well as to avoid re-engineering business processes. Additionally, this design helped simplify the usage of the system for the end-user because it resembles the pre-existing system.

The customer table includes the fields Customer ID, First Name, Last Name, Address, City, State, ZIP, Phone Number, E-mail Address, and Notes. The majority of these fields are straightforward in terms of content and are simple string data types, but a few call for special attention.

Customer ID was one of the few fields created exclusively for use in the database without a counterpart in the existing paper documentation. It consists of a relatively simple auto-numbered integer value which uniquely identifies each customer.

The Notes field consists of freeform alphanumeric information about a customer. This field was requested by the owner of Lightspeed specifically and existed in the original paper forms. Notes is a string field with a length of 255 characters, allowing the owner to create quite lengthy notes about a customer should the need arise. The creation of a Notes table was considered as an alternative to this field, but the owner indicated it was not necessary to maintain multiple notes for each customer, rendering this additional database complexity useless.

The transaction table consists of the fields Transaction ID, Customer ID, Date, Hardware Value, Software Value, Labor Value, Tax Value, Down Payment, Paid In Full, and Notes. The Date field stores the date of the transaction in date/time format. The 'Value' fields and the Down Payment field are all numeric fields containing decimal data representing dollar values.

Transaction ID, like Customer ID in the customer table, is a relatively simple integer auto-number used to uniquely identify each transaction. The Customer ID in the transaction table functions as a foreign key, relating a transaction to a record in the Customer Database. Both of the ID fields did not have any precedent in the paper documents already in use at Lightspeed, but are necessary for an RDBMS system.

Paid In Full is a relatively simple boolean field representing the payment status of a transaction. Lightspeed does not accept partial payments, so a boolean field is sufficient to track the payment status. It is worth noting, however, that the Down Payment field could serve as a way to track partial payments should Lightspeed later decide to change their policy regarding partial payments.

The Notes field is very similar to the Notes field in the customer table, although in this case the Notes are connected to individual transactions instead of individual customers. In all other aspects the transaction Notes field is identical to the Notes field in the customer table, including design motivation.

Once the database design was finalized, the database was populated with existing Lightspeed customers and some existing transactions. However, due to the owner's desire for both customer privacy and data security, none of the data from that database will be presented in this documentation.

VIII. VB.NET Implementation

Microsoft VB.NET is a very good choice for rapid development of Windows-based office applications. VB.NET includes many database objects which can be used to add, edit, or view records in almost any kind of database. In addition, Visual Studio 2003 includes many tools to facilitate the development of database applications. All forms in this project used a combination of an OLEDB Connection, an OLEDB Data Adapter, and a Dataset (Holzner, 2003).

The OLEDB Connection provides the connection to the database. In this case, the database was made in Access, so the OLEDB Connection object uses Microsoft Jet. The OLEDB Connection object only provides a 'path' which allows a form to access the database, but another object is needed to actually perform that access.

The OLEDB Data Adapter is the intermediary that allows a form to retrieve data from a connected database. The OLEDB Data Adapter is what stores any SQL statements that a form needs to execute while the OLEDB Data Adapter's methods actually execute the SQL statements. The OLEDB Data Adapter is also responsible for adding or editing records within the database, although this function also relies on another intermediary.

The Dataset is the data that the form works with directly. The Dataset is filled by the OLEDB Data Adapter, after which the Dataset is used to access the data in the database. Any alterations to data are also performed in the Dataset, but the OLEDB Data Adapter is responsible for committing those changes to the database.

These three objects are responsible for most of the work in all database-connected forms. The connection objects in all forms are identical, but there are severe differences

in the data stored in the Datasets as well as the queries used in the OLEDB Data Adapters. Different forms require different data, and the SQL statements in the OLEDB Data Adapter reflect these differing desires, as do the Datasets.

Another common element to all of the VB.NET forms as well as the ASP.NET forms is the lack of a delete button or function of any kind. The owner of Lightspeed, late in the planning process, expressed a desire to not have delete functionality in the interfaces. This desire was driven by a fear of accidental record deletion as well as concerns about possible misuse by any future employee. Currently, the only way to delete a record is by manually editing the database in Access, something the owner is comfortable with. It would be relatively easy to add an authenticated deletion page to both interfaces while still maintaining security if the owner desires the addition at a later time.

Data Validation is supported by a single module which contains several basic validation functions. Most of the validation is handled by regular expressions, although some is provided by built-in VB.NET functions. Individual forms also contain validation code which calls the correct validation functions and handles the output of error messages.

The aforementioned implementation details provide a limited overview of the project, but there are enough specific implementation details available to justify the following form-by-form analysis of the project, starting with the View Customers form.

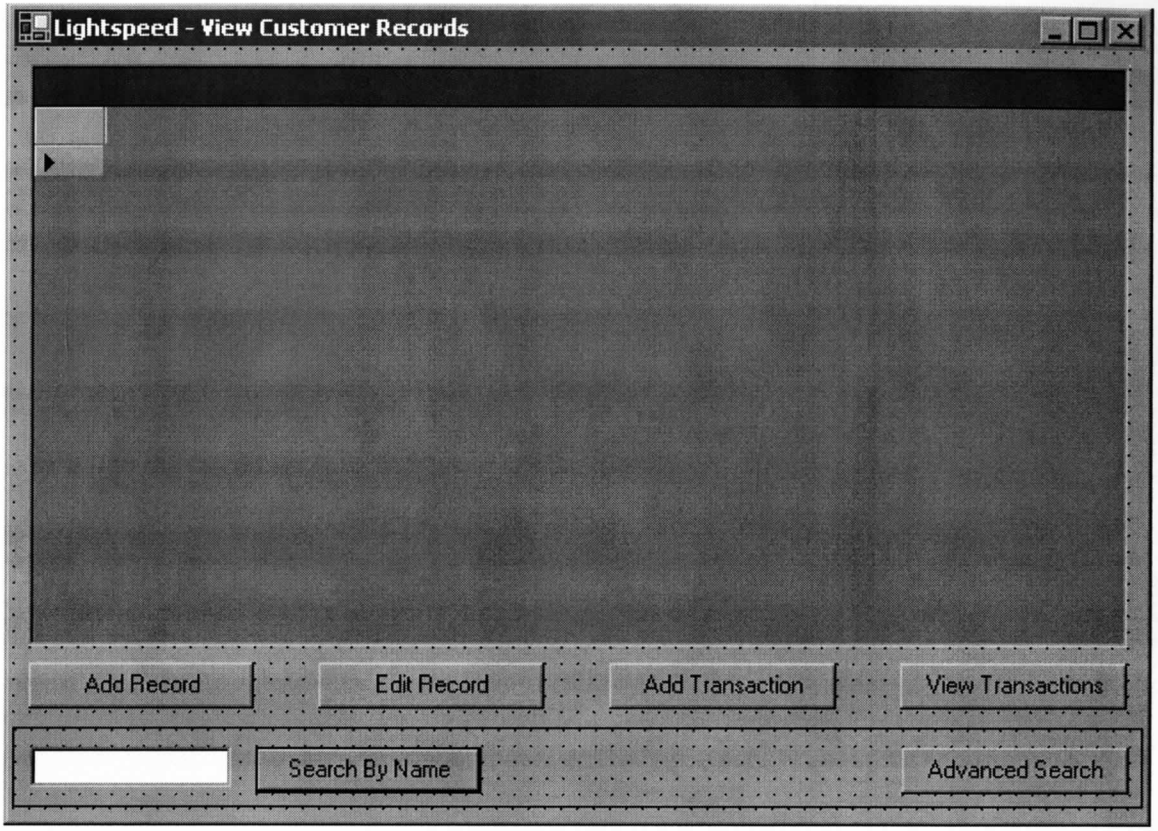


Figure 8.1: The View Customers Form

The View Customers form is dominated by the large DataGrid control. The DataGrid control is used to display the information stored in the Dataset. The Dataset is, by default, populated with all of the customer records in the database. However, the content of the dataset can easily be altered as necessary.

The easiest way to filter the records that are show in the DataGrid is by using the "Search by Name" button on the bottom of the form. This button adds a WHERE clause to the SQL Statement, using the text entered in the adjacent textbox as a comparison value to find records with a similar First Name or Last Name. This allows the user to quickly filter the results to more easily find the desired customer record. Also, the "Advanced Search" button can allow the user to filter the results shown in the DataGrid, but that will be examined in more detail later.

The "Add Record" and "Edit Record" buttons function very similarly to each other, redirecting the user to the Add/Update Customer form. The "Edit Record" button also retrieves the currently selected Customer ID from the DataGrid and uses that to populate the Add/Update Customer form with the initial data.

The "Add Transaction" button functions similarly to "Edit Record", redirecting the user to the Add/Update Transaction form. "Add Transaction" also retrieves the currently selected Customer ID, which is then used by the Add/Update Transaction form to create the new transaction record.

"View Transactions" redirects the user to the View Transactions form and also retrieves the currently selected Customer ID and uses that to filter the transactions that are shown.

This form is also designed to show custom queries generated by the Advanced Search form. This is accomplished by changing the query stored in the OLEDB Data Adapter to retrieve the appropriate records when the user is directed to the View Customers form by the Advanced Search form.

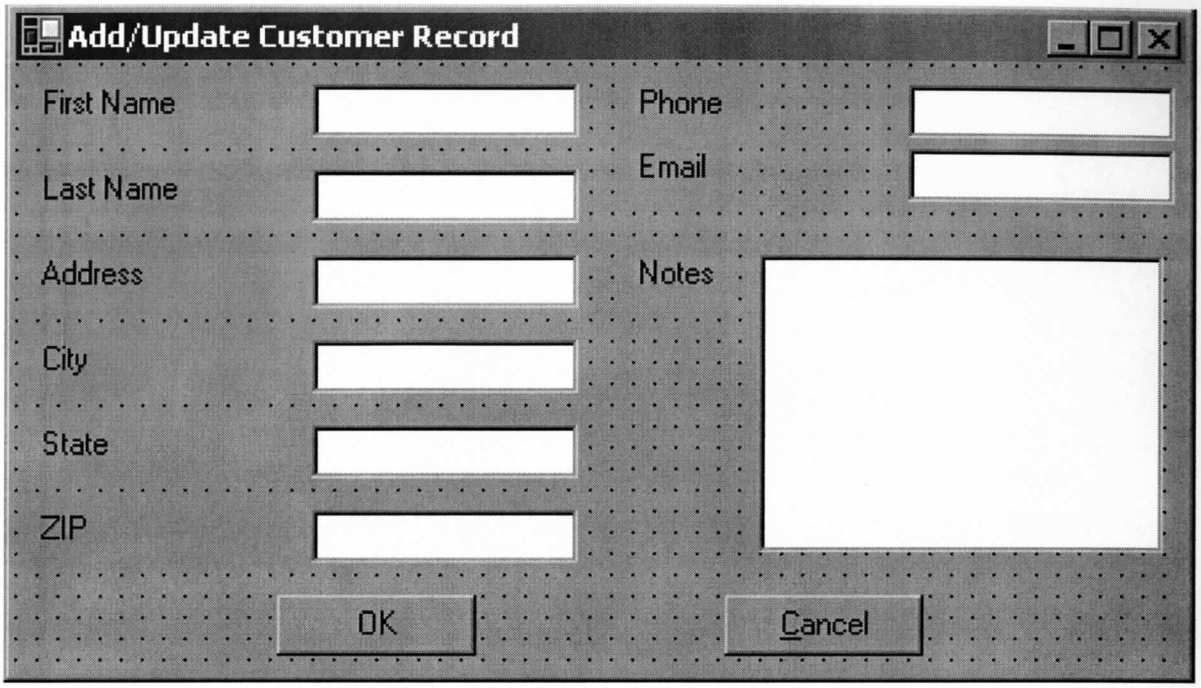


Figure 8.2: The Add/Update Customer Form

This form contains textboxes for all fields in the Customers table with the exception of Customer ID, which is not user-editable in any way and will be auto-generated when a new record is created. The form is fairly straightforward, but there are some aspects that require explanation.

The development of this form was aided by the Data Form Wizard. The Data Form Wizard creates a basic form with bound controls to display or input all desired fields from a table. In order to bind these controls to the data, the wizard creates an OLEDB Connection, an OLEDB Data Adapter, and a Dataset. In addition, the wizard creates several functions designed to update the database with any changes that are made to the Dataset, including the addition of new records, deletion of old records, and modifications to existing records.

Unfortunately, while the Data Form Wizard provides a great base of code to work from, the code it creates was not directly compatible with the design of this project. As a

result, the code was significantly modified to support the flow of the design. One major change was loading the appropriate data on page load when necessary instead of waiting for the user to click a button.

Another major change to the functionality of the Data Form Wizard was the elimination of several command buttons. Normally the Data Form Wizard assumes that a user will be able to navigate the database freely within the form, whereas the project design called for navigation in a separate form. The most obvious result of this conflict was the removal of the navigation controls from the basic form.

Another consequence of the lack of navigation resulted in the removal of several buttons entirely, including a "Cancel changes to this record" and "Add" button. "Cancel changes to this record" was removed entirely because there is no need to cancel changes to an individual record because there will only be a single record being added or edited at any time, making this and "Cancel all changes" identical in functionality. The "Add" button was unnecessary because the user chooses to add a record before they see this form, rendering it useless at best and redundant at worst. Instead, if a user was directed to the form by an "Add Record" button, the page load event detects that and executes the code that was associated with the "Add" button automatically.

As shown in Figure 8.2 above, the final form only has the buttons "OK" and "Cancel". The "OK" button submits the changes or additions on the form, updates the database with the changes, and returns to the View Customers form. The "Cancel" button is identical to the "Cancel All" button generated by the Data Form Wizard except that it also returns the user to the View Customers form.

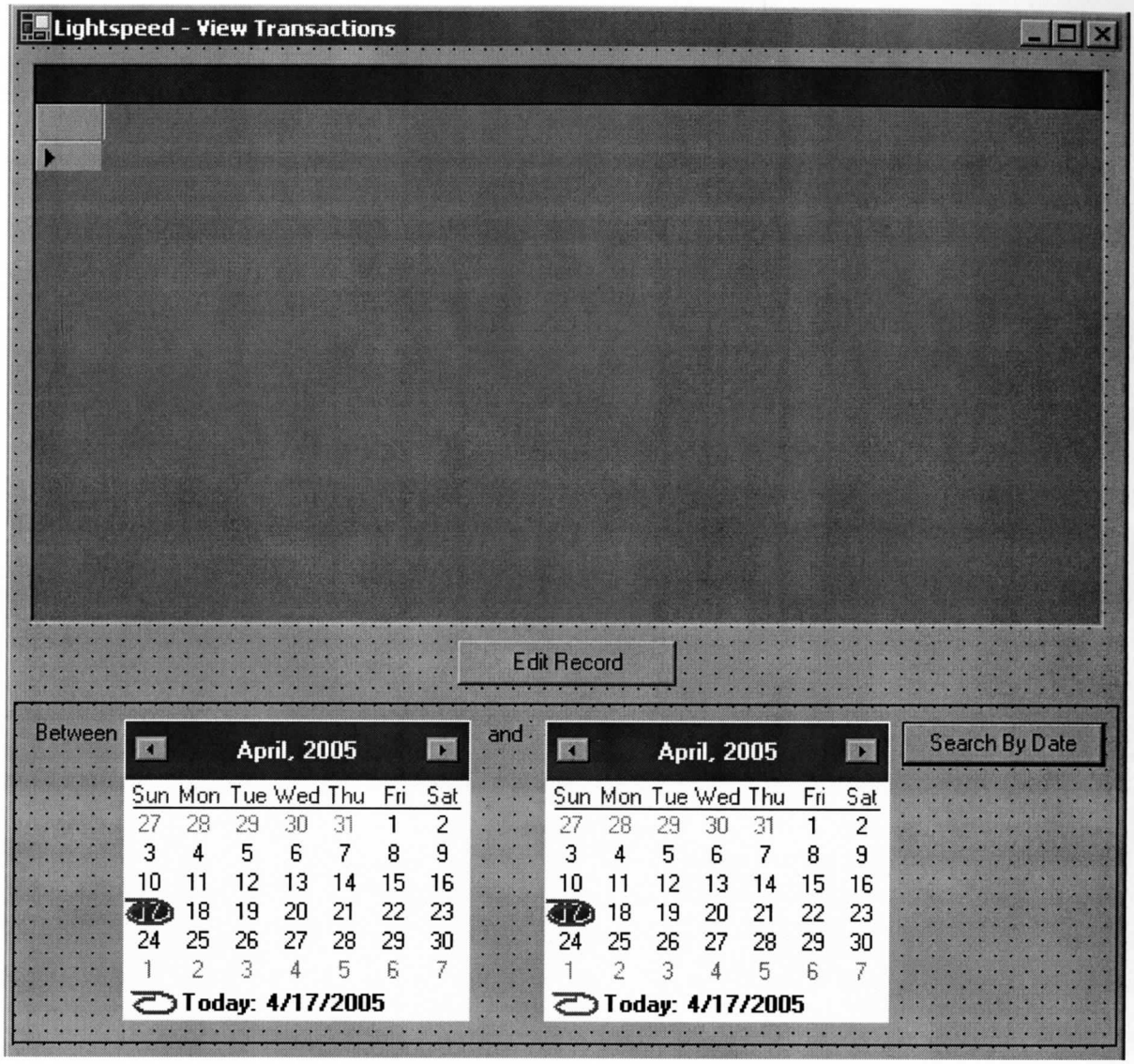


Figure 8.3: The View Transactions Form

The View Transactions form is very similar to the View Customers form in many ways. In fact, the View Transactions form began as a copy of the View Customers form, so much of the code is nearly identical. This form is similarly dominated by a large DataGrid, although it is obviously loaded with data from the transactions table instead of the customers table.

The "Edit Record" button functions similarly to the "Edit Record" button in the View Customer form. When it is clicked, it redirects the user to the Add/Update

Transaction form, sending the form the Transaction ID of the currently selected item in the DataGrid. This Transaction ID is used to populate the Add/Update Transaction form.

The "Search by Date" button is similar in functionality to the "Search by Name" button in the View Transactions form, but the implementation is quite different. Most noticeable is the inclusion of two Calendar controls to facilitate selecting a date for use as a search term. These Calendar controls were used because textboxes are problematic for Date inputs due to the relatively strict nature of the date/time data format. The use of the Calendar controls alleviated worries about input data type errors, making the search much more reliable from the user's perspective. When "Search by Date" is selected, the OLEDB Data Adapter is updated with a new SQL statement, the Dataset is refilled, and the DataGrid reflects those changes.

The View Transactions form was also designed to provide output for the Advanced Search form, accomplished by altering the query used in the OLEDB Data Adapter. The development of the Advanced Search form will be covered in more detail later in this document.

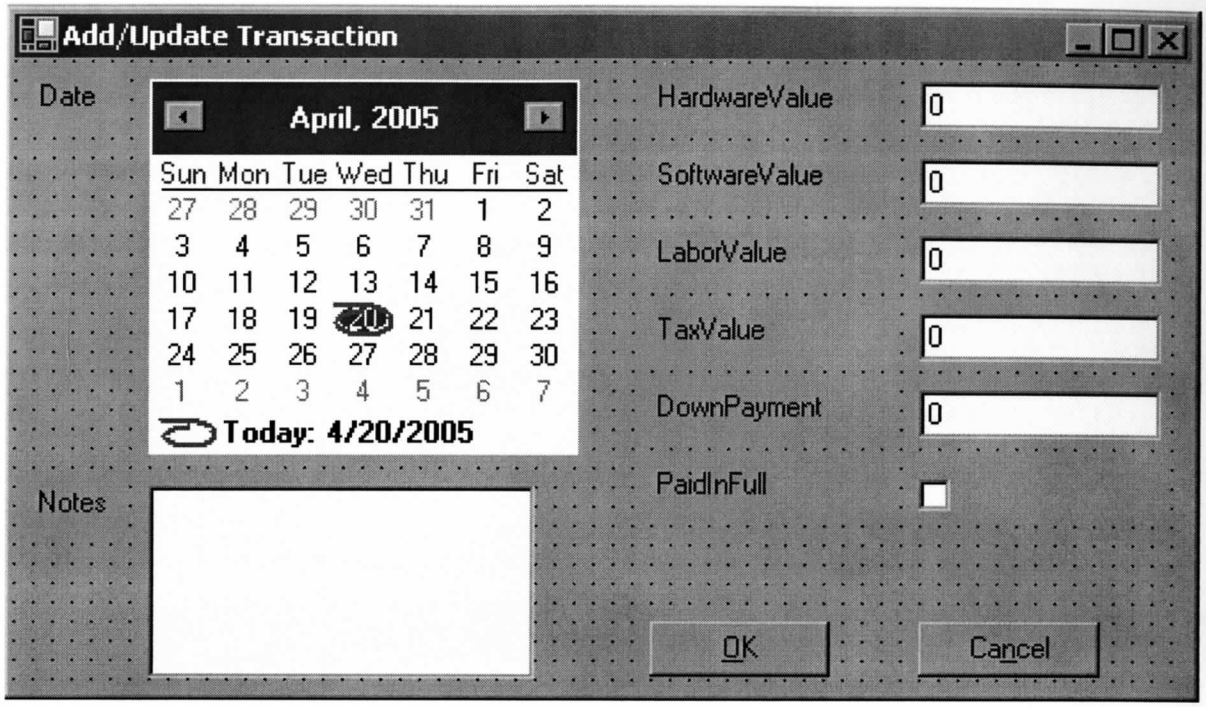


Figure 8.4: The Add/Update Transactions Form

Just as the View Transactions form was similar to View Customers, the Add/Update Transactions form is very similar to the Add/Update Customers form. Again, this form was created using the Data Form Wizard in Visual Studio 2003. Many of the manipulations to this form in order to make it flow correctly are identical to the changes made in the View Customers form. Since these changes are so similar, they will not be outlined individually here.

Most of this form is relatively self-explanatory. It consists mostly of textboxes used to display or input information in the database. The boolean field Paid In Full is displayed as a checkbox, the most efficient way to illustrate a boolean data type in a graphical form.

The only major design change is the inclusion of a Calendar control for use with the Date field. Much like the View Transactions form, this was utilized to decrease the likelihood of input difficulties. If a new record is being added, the Calendar control

displays the current date; if a record is being edited, the Calendar control displays the date stored for that record in the database.

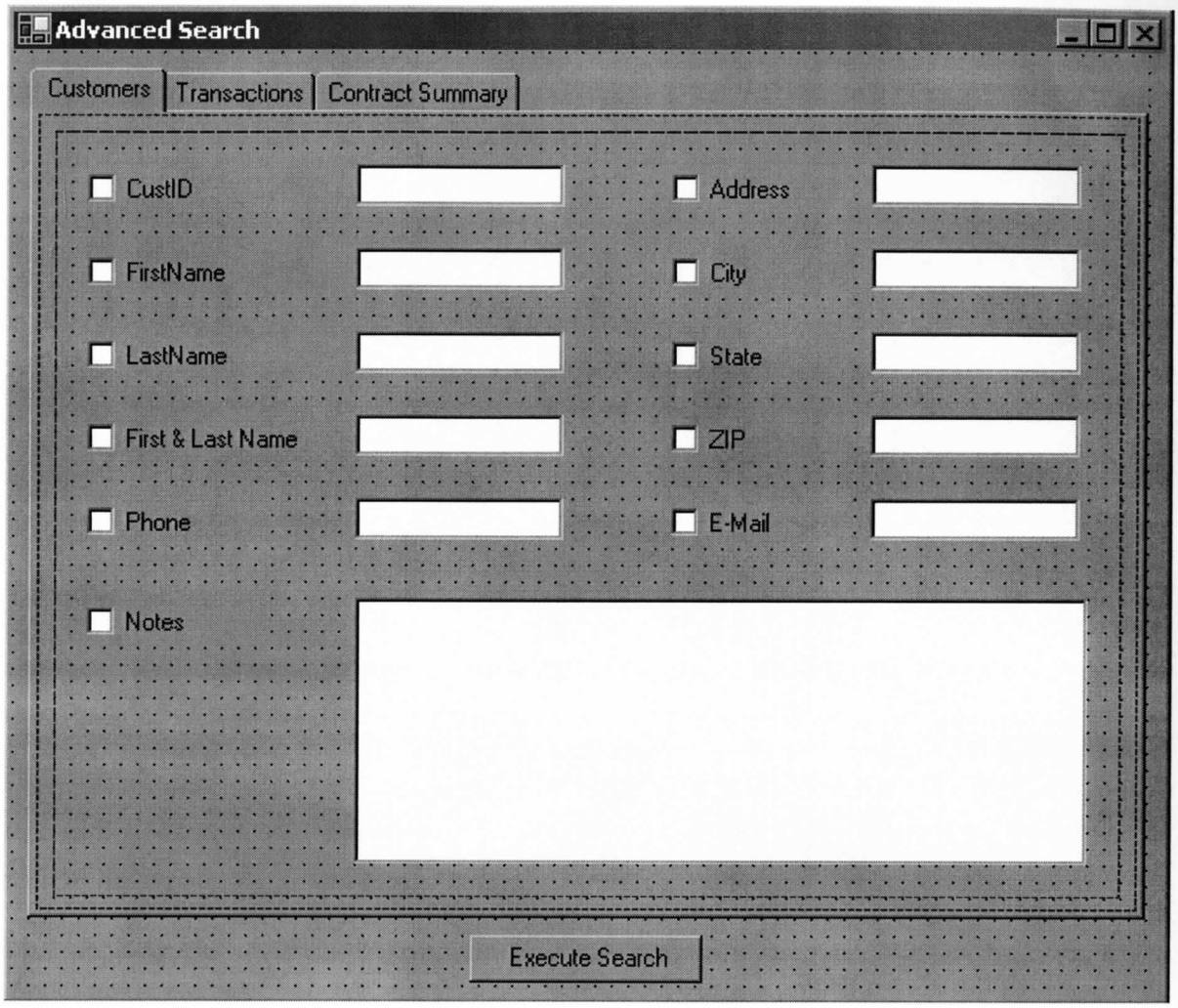


Figure 8.5: The Advanced Search Form, Customers Tab

This form is a fairly straightforward data-entry page, sharing many elements with the aforementioned Add/Remove Customers Form. The purposes of the textboxes should be reasonably obvious given the labels being used – they are used to provide search terms for their respective fields.

The checkboxes beside each textbox/label pair determine whether any data entered in the textbox is actually used for the query and validated. If the box is

unchecked, the query generating code and validation will ignore any text in the related textbox. Some checkboxes perform other special functions: for instance, checking the "FirstName" and "LastName" checkboxes will uncheck the "First & Last Name" checkbox to avoid conflicts, while checking "First & Last Name" will uncheck both "FirstName" and "LastName" for the same reason.

The query-generating code is fairly simple but lengthy, consisting primarily of several conditional statements to check the status of the checkboxes as well as the validity of data. The query string is initialized to an appropriate base query: in this case SELECT * FROM CUSTOMERS. If a given checkbox is checked and the data in its associated textbox is valid, the query generator appends appropriate SQL to the end of the initial string in order to achieve the desired result. The end result is a properly-formed SQL Statement which can then be used in an OLEDB Data Adapter to load the desired records.

The "Execute Search" button is responsible for executing the correct query generation code, passing on the search query, and redirecting the user to the appropriate display form. The correct code path is selected by an IF statement which uses the currently selected tab as a test condition.

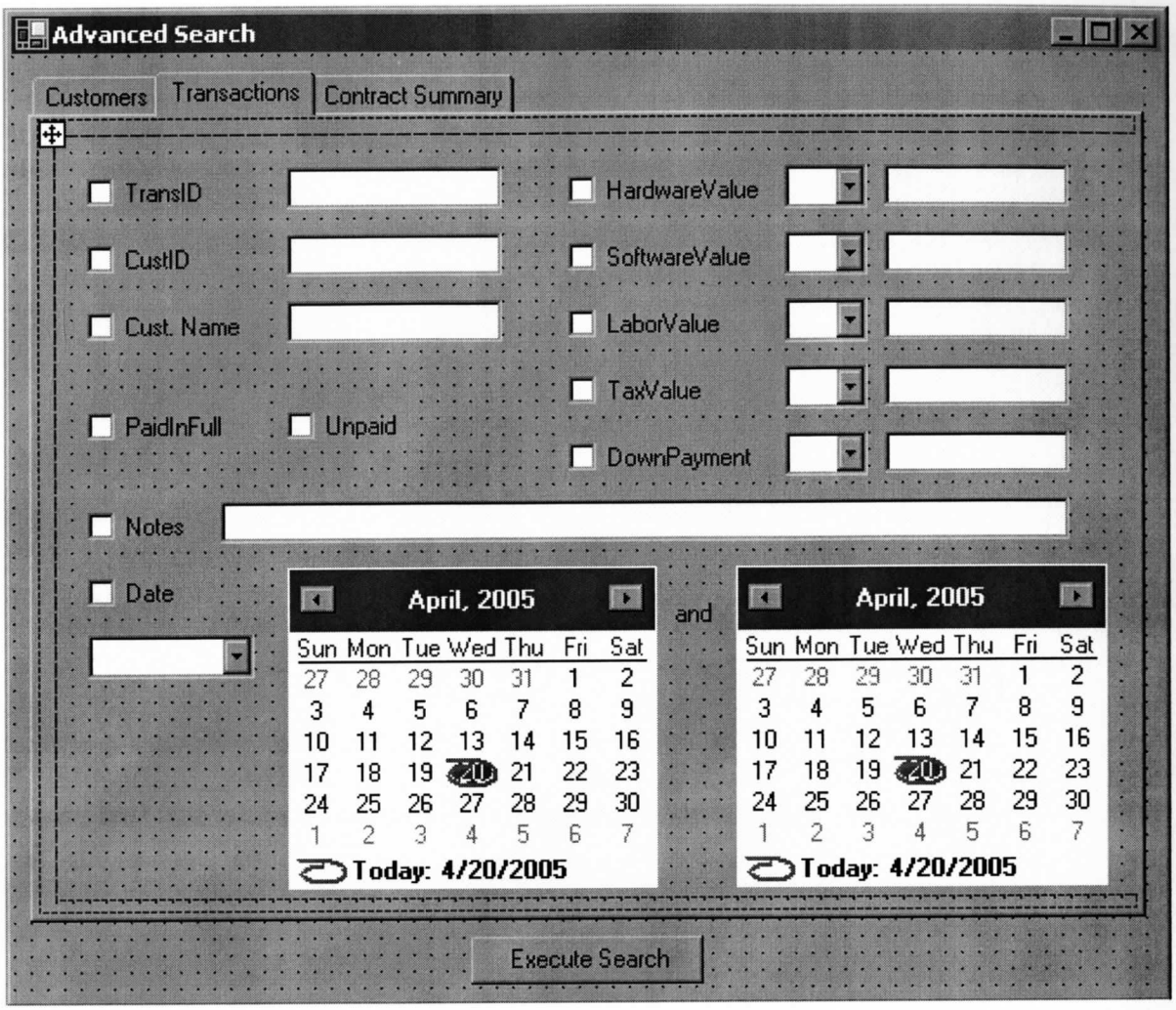


Figure 8.6: The Advanced Search Form, Transactions Tab

Much of this form resembles the Add/Update Transaction form. Most of the textboxes and checkboxes are obvious in terms of content and the query builder functions very similarly to the Customers tab of the Advanced Search form.

There are several design features unique to this tab which are worthy of note, most obviously the comboboxes (or drop-down list boxes) used to aid query generation. The comboboxes present between the checkboxes and textboxes for the monetary values (HardwareValue, SoftwareValue, LaborValue, TaxValue, and DownPayment) allow the user to select the operator used for comparison by the query engine. The combobox

allows the user to choose equal, greater than, less than, greater than or equal to, and less than or equal to operators as desired. This obviously adds some complexity to the query generator, but it does not have an overly large effect.

Another combobox of note is underneath the "Date" checkbox. This combobox allows the user to choose either "Equal" or "Between" as a date-search operator. If "Between" is selected, an additional calendar control (visible in the above screenshot) is made visible to the user to allow the selection of a range of dates. Similar to the aforementioned comboboxes, this also added some complexity to the query generator.

The only other unique design feature involves the PaidInFull checkbox. In order to make the search terms more clear to the user, when the data value checkbox is checked it will display "Paid" and when it is unchecked it will display "Unpaid". This was an easy-to-implement feature suggested during testing to improve usability.

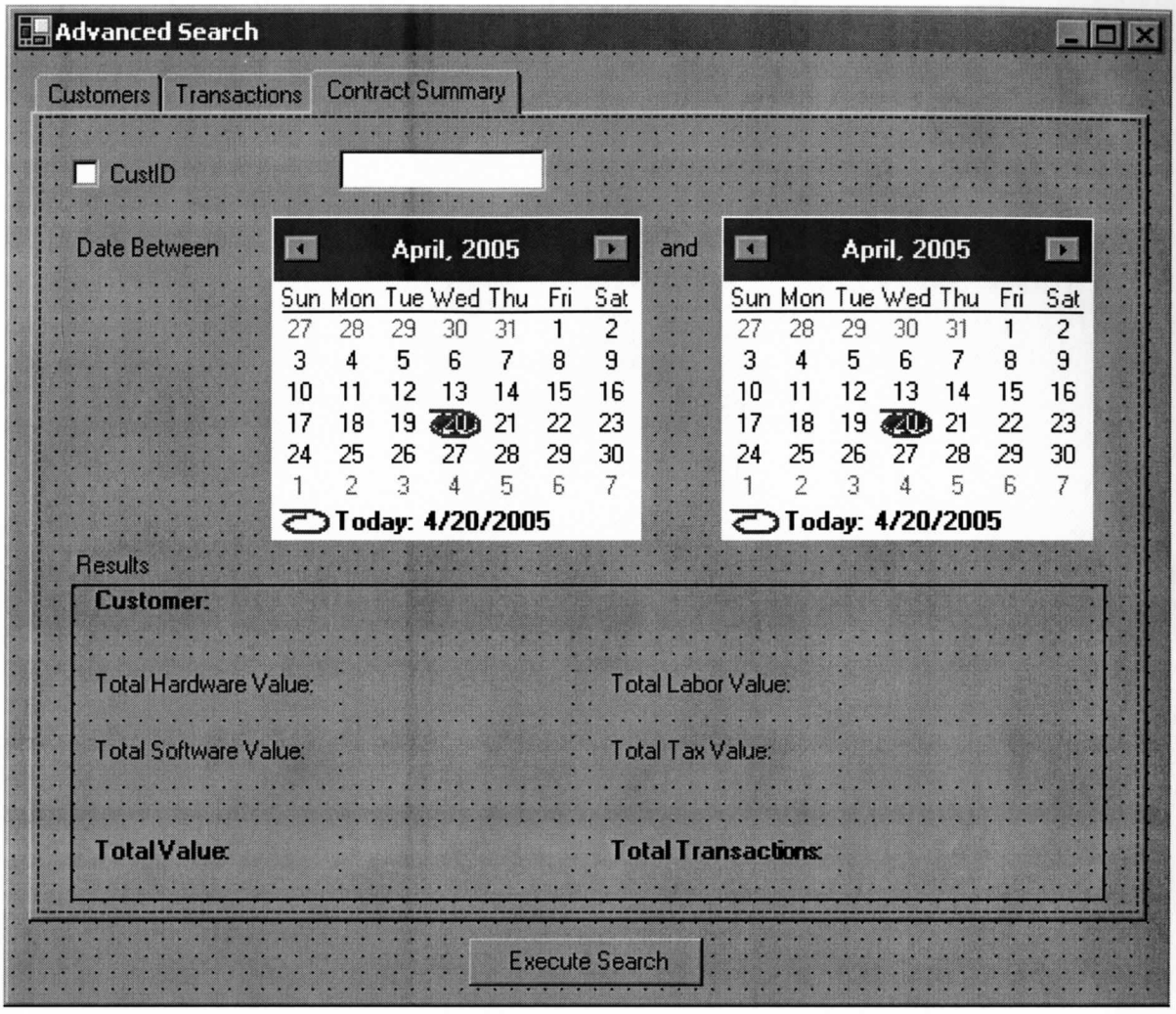


Figure 8.7: The Advanced Search Form, Contract Summary Tab

The Contract Summary tab is quite different from the other search tabs. Aside from the relative simplicity of the interface, the output for the searches performed on this tab is handled by the labels in the "Results" panel at the bottom of the tab. This search is designed to allow a user to specify a period of time and get summary information about that period. The user may optionally specify a CustID to retrieve summary information for only one client, presumably a contractual client.

This tab has a very simplistic version of the query generator used on the other two tabs, choosing between two general queries based on the status of the CustID checkbox.

When execute search is clicked, an OLEDB Connection is established, an OLEDB Command is created, and an OLEDB Data Reader is created. The OLEDB Connection is used by the OLEDB Command to execute the generated query and feed the results to the OLEDB Data Reader. The Data Reader is then used to fill the labels in the "Results" area.

IX. ASP.NET Implementation

The ASP.NET web interface was originally planned to use much of the same code as the VB.NET interface. Unfortunately, differences in the implementation of the DataGrid in the two formats, as well as some other minor differences, prevented the reuse of much of the VB.NET code. Even though the programming language is essentially the same, these small differences require some significant alterations to the code.

Unfortunately, development of the ASP.NET interface began much later than originally planned. This late start, combined with the differences between VB.NET and ASP.NET, caused the elimination of some features. However, the features that were removed were the search features on the View Customers and View Transactions panels, which were mentioned as potential removals in the original project plan.

Also, the compressed timeframe somewhat eliminated the planned prototyping methodology. While the development still roughly followed a prototype methodology, there was not as much testing of each prototype as originally planned. Instead, the testing period was very brief and a new prototype was created as soon as basic functionality was verified.

Development of the ASP.NET code was performed in Web Matrix, a free development tool distributed specifically for creating ASP.NET pages. Web Matrix did not have the wizards that make the implementation of the OLEDB Connection, OLEDB Data Adapter, and Dataset trio so easy in VB.NET.

Instead, the ASP.NET code utilized a combination of the OLEDB Connection and OLEDB Command objects to display and update the database. The OLEDB Connection

object is identical in purpose to the OLEDB Connection object in VB.NET, providing only the connection with no methods to manipulate the data in any way. (Walther, 2004)

The OLEDB Command object performed data retrieval, insertion, and updating via SQL command strings. These commands ranged from very simple SELECT statements to reasonably complex INSERT and UPDATE statements involving the usage of parameters. In most cases, the SELECT statements were used to fill the DataGrids in the View Customers and View Transactions panels. (Walther, 2004)

Data validation code remained largely unchanged from the VB.NET implementation, although in ASP.NET it is present as a code-behind file instead of as a module. The query-generating code remained largely unchanged as well, although the actually execution of the query is significantly different.

The ASP.NET interface consists of only 3 active pages and 1 code-behind source file. View Customers, View Transactions, and Advanced Search were all condensed into a single page, eliminating many potential problems involving communication between those pages as well as several security concerns about including large amounts of information in the page URL. This did not seem to have a significant adverse effect on the performance of the interface, although Web Matrix had difficulties rendering all of the elements.

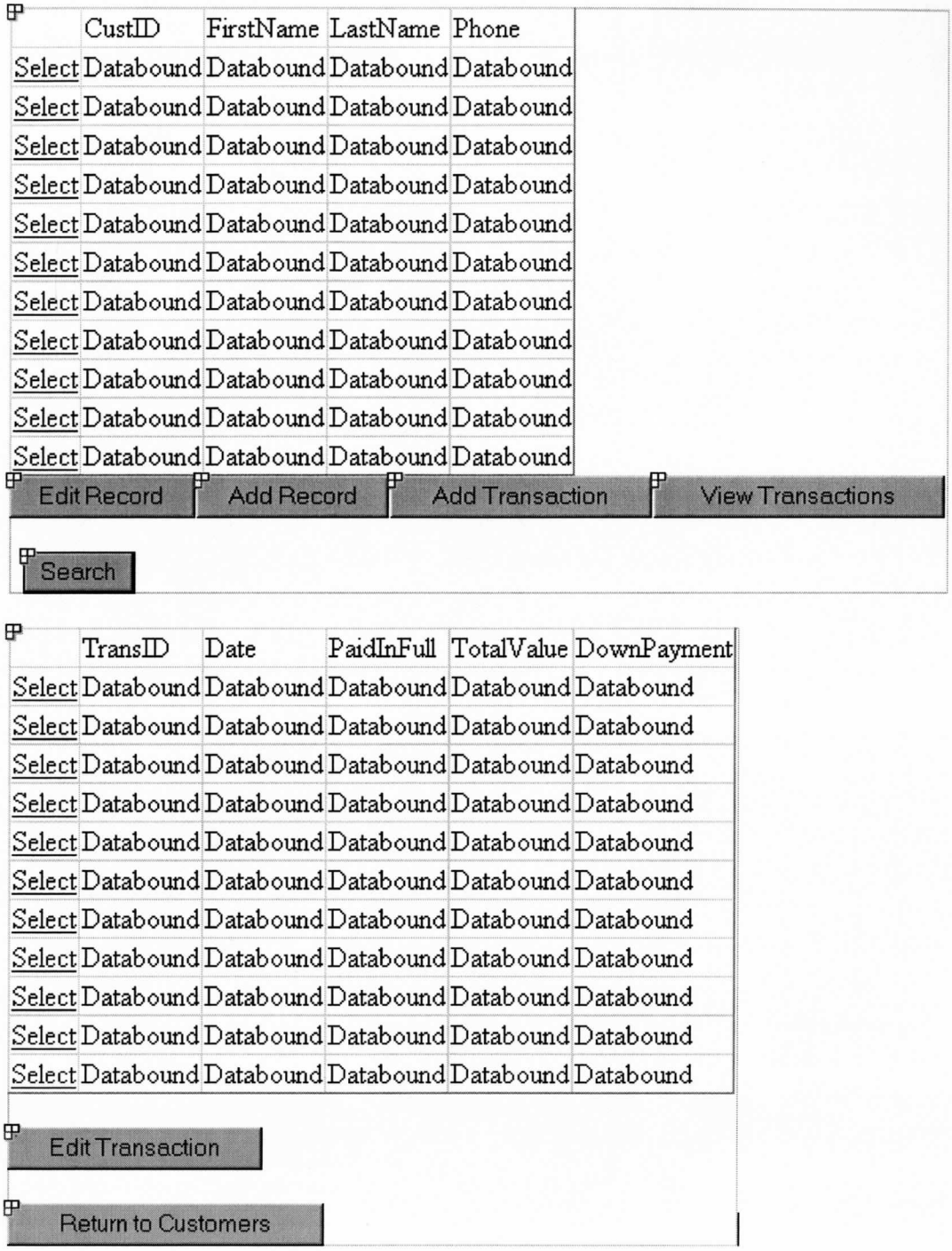


Figure 9.1: View Customers and View Transactions Panels

Unlike the VB.NET implementation, the ASP.NET implementation unifies View Customers and View Transactions into a single page. These two separate functions are

instead handled by forms within the two pages. This facilitates communication between the two views, allowing them to more easily influence each other.

The major difference between the View Customers ASP.NET panel and the VB.NET form is the lack of the search button in the ASP.NET code. Otherwise, the two interfaces are quite similar to each other. The buttons function nearly identically, although enough differences exist that it is worth discussing them.

The "Edit Record" and "Add Record" buttons pass information similarly to the VB.NET equivalents. However, instead of calling a separate form, these buttons redirect the user to the Add/Update Customer ASP page, passing information to that page using query strings appended to the URL of the page. The "Search Button" serves only to open the Search Panel.

The "Add Transaction" button functions similarly to both the aforementioned "Edit Record" and "Add Record" buttons as well as the "Add Transaction" button in the VB.NET implementation. The ASP implementation of the "Add Transaction" button redirects the user to the Add/Update Transaction ASP page. Information is passed to that page using query strings appended to the page's URL.

The "View Transactions" button doesn't call a separate page; instead it makes a panel within the same page visible while making the "View Customers" page invisible. It also triggers a function to update the DataGrid in the relevant panel. The button click also triggers the storage of some information to invisible label controls used to share information between the panels.

The View Transactions panel includes an "Edit Transaction" button which functions similarly to the "Edit Record" button in the VB.NET interface. When the

button is clicked, it redirects the user to the Add/Update Transaction page, passing the selected Transaction ID and Customer ID by appending variables to the URL.

One major difference in the usage of the ASP.NET interface over the VB.NET interface is an effect of the differing implementation of the DataGrid in ASP.NET. Since the DataGrid does not allow selecting individual items by default, an extra column was added which allows ASP.NET to mimic this functionality with a button click. This column of buttons is quite important to the functionality of the ASP.NET interface, and was very important to maintaining any kind of commonality between the ASP.NET and VB.NET implementations.

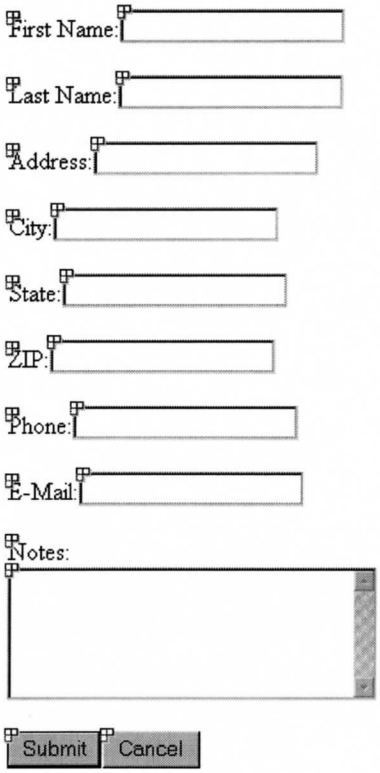


Figure 9.2: Add/Update Customer ASP.NET Form

The Add/Update Customer ASP.NET implementation is completely different then the Add/Update Customer VB.NET implementation. The code was completely rewritten

due to the absence of a Data Form Wizard in ASP.NET. Functionally, however, the ASP.NET and VB.NET forms are identical.

The textboxes serve as both input and output for the form, loaded by an OLEDB Command object containing a simple SELECT statement when used for editing. When the form is used to add a record, the textboxes initially have no data. When data is entered into the form, the code generates an INSERT command by assigning values to parameters created in another OLEDB Command object.

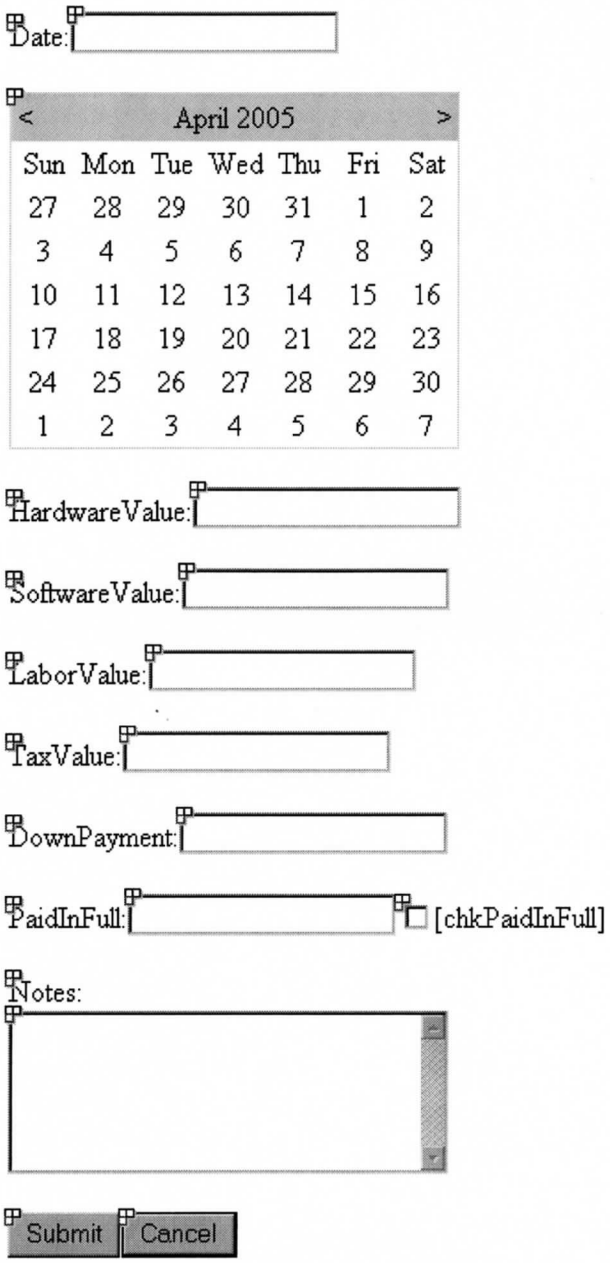


Figure 9.3: Add/Update Transaction ASP.NET Form

The Add/Update Transaction ASP.NET form is very similar in implementation to the Add/Update Customer ASP.NET form and very similar in functionality to the Add/Update Transaction VB.NET form. The implementation details are so similar to the Add/Update Customer ASP.NET form that any further explanation of this form is largely redundant.

One detail worth discussing is the Calendar control. Implementation of the Calendar control was significantly different then in VB.NET, resulting in the inclusion of a textbox for early testing purposes. That textbox is still in the code for the page, although it will never be visible to the user, and is used as an intermediary between the database code and the Calendar control. Similarly, the checkbox for PaidInFull has an intermediary textbox.

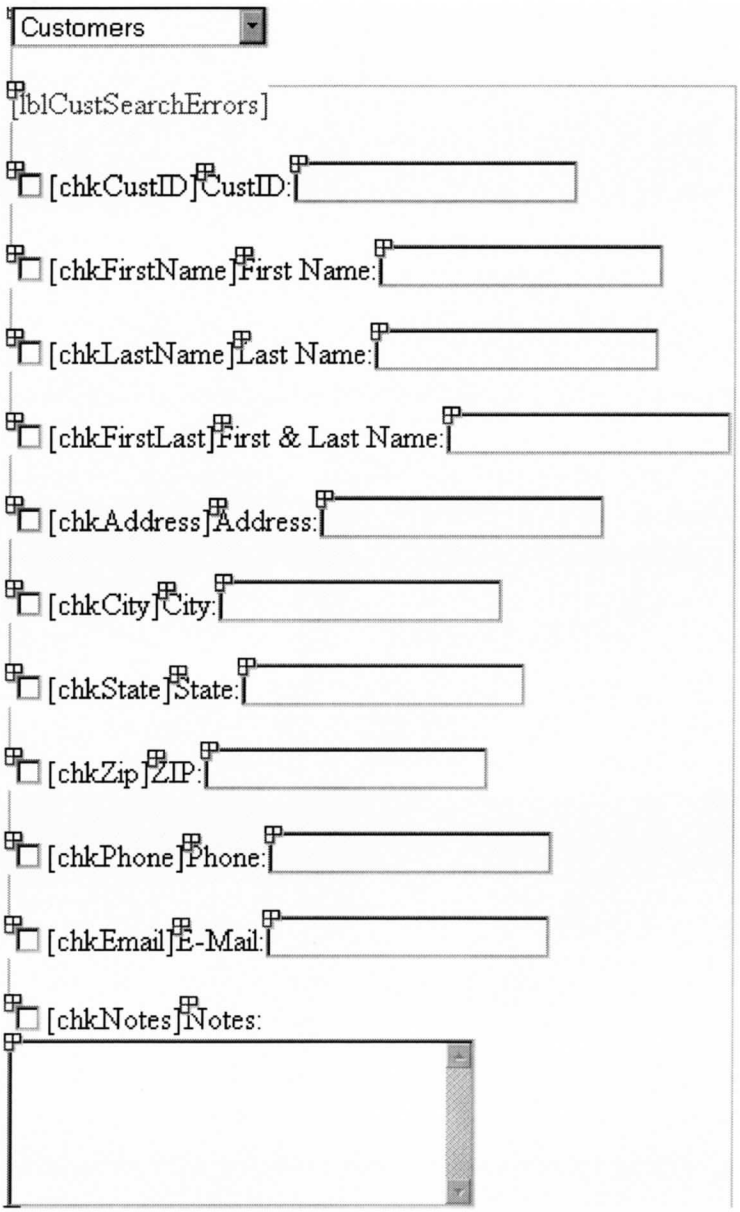


Figure 9.4: Customer Search Panel

The Customer Search panel is very comparable to the Advanced Search form's Customers tab, and in fact shares much code with that tab. The query generator is almost identical to the VB.NET equivalent, and the error handling code is also very similar.

Errors are displayed in the red label at the top of the panel, which is only visible when an error has been raised. Also of note is the drop-down list box at the top of the form. This drop-down list is the control which allows users to select the type of search they want to perform, allowing access to this and the other two search panels.

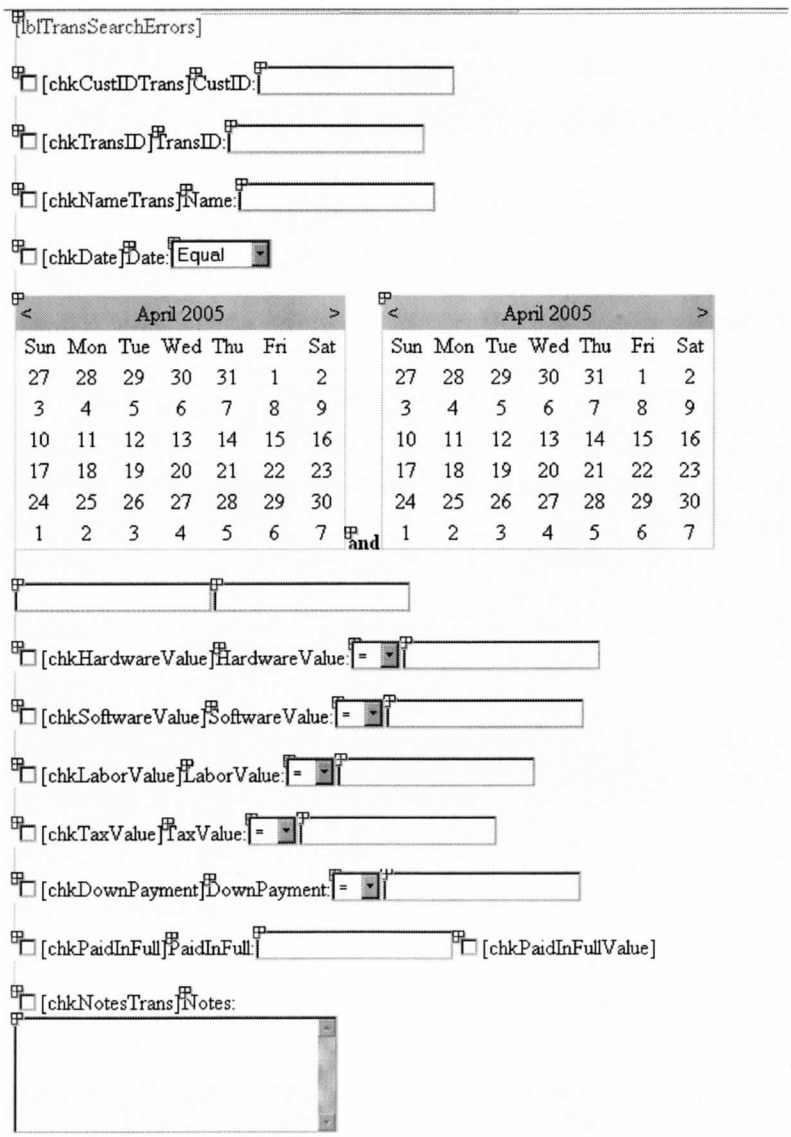


Figure 9.5: Transaction Search Panel

The Transaction Search panel is also quite comparable to its VB.NET counterpart, and is largely self-explanatory. Most of the unique design features are also present in the VB.NET version and were already addressed, so functional analysis would be redundant.

There are some features worth individual mention, however. This form has intermediary textboxes like the Add/Update Transaction ASP.NET form: one for each Calendar control and one for the PaidInFull checkbox. Also of note, the drop-down list boxes in this form are identical in functionality to the comboboxes in the VB.Net interface, but their actual implementation was quite different. The final object of note is the red textbox at the top of the interface, used for error output similarly to the red textbox in the Customer Search panel.

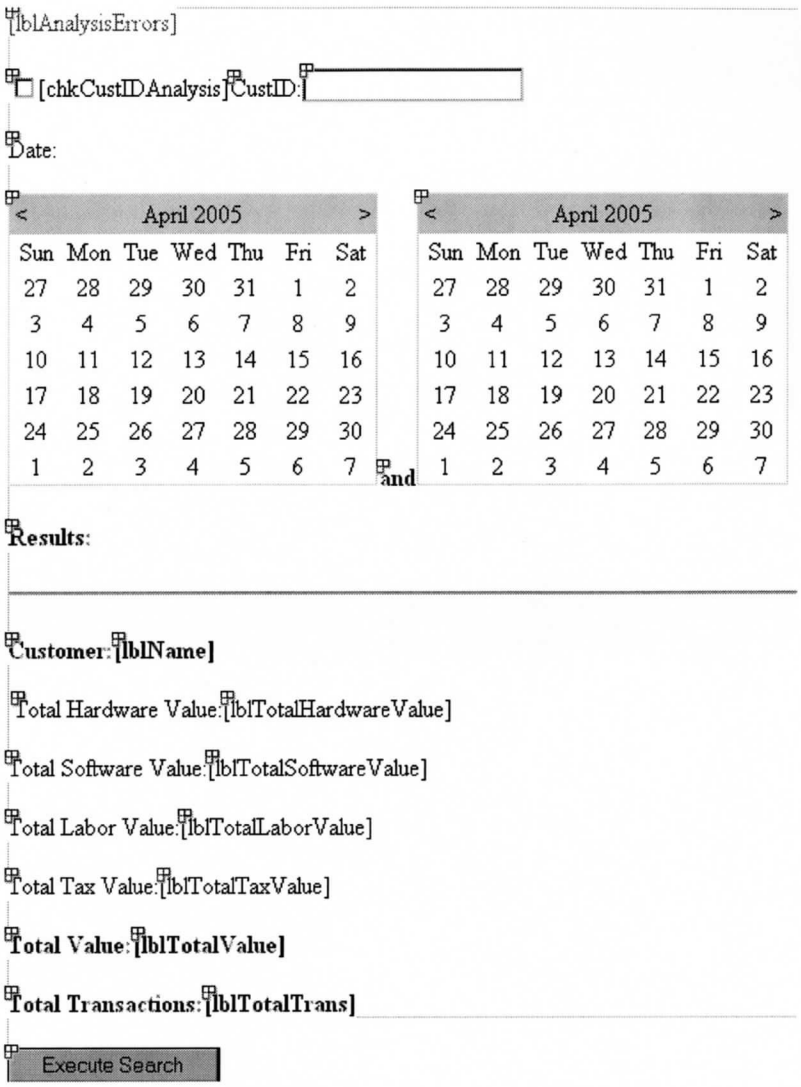


Figure 9.6: Contract Summary Panel

This form is functionally identical to its VB.NET counterpart, and analysis of that functionality would be redundant. There are no controls unique to this panel that are worthy of much additional commentary, although it is worth noting that there is a red label which is used to output error messages.

The most important element in this screenshot is the "Execute Search" button, which is actually a functional part of all Search panels and is responsible for executing the correct query generators according to the currently selected item in the main Search

panel drop-down list box. This button functions similarly to the equivalent button in the VB.NET interface, but there are some important differences. The primary difference is that this button isn't passing query information to another form for display purposes. Instead it simply stores query information, executes a procedure to fill the proper DataGrid, and makes the proper panel visible.

In the case of a Contract Summary search, the button does not fill a DataGrid or alter panel visibility. In this case, the button executes a procedure which queries the database and displays the results to the label controls on the Contract Summary panel itself. The code behind this procedure is identical to the code described for the Contract Summary tab in the VB.NET interface.

X. Results and Conclusion

The Lightspeed Technologies database and interfaces meet the requirements and objectives set for them by both project planning and the owner of Lightspeed. Valuable customer information is readily available and searchable. Transaction information is likewise readily available and searchable. The interfaces also support the retrieval of summary information useful in the evaluation of service contracts.

The owner of Lightspeed reports that both front ends are quite easy to use and practical for day-to-day usage. Information can be added and manipulated within the database easily, and the interfaces are familiar enough to Windows users that any potential employee should not have difficulty learning to use them effectively.

The system is currently being utilized as anticipated. Currently, transaction information is not being regularly entered into the system because the owner only desires transaction information for contractual clients. The customer information, however, is being used regularly for various purposes. The owner expressed appreciation for the Notes fields within the two tables because it helps him keep track of information that he had been keeping track of mentally until the implementation of this system. This should also help smooth the process of hiring a new employee in the future.

Unfortunately, the future could hold many potential hurdles for this system. If Lightspeed wishes to utilize the ASP.NET front-end from a remote location, the Access database will need to be migrated to something more secure in order to support that operation. Also, the database will need a new table to contain login information, and the ASP.NET front-end will need an authentication page to utilize this information. Luckily, this migration should be relatively painless. The code of both the ASP.NET and

VB.NET front-ends only requires changes to OLEDB Connection objects' connection strings to be functional on a new DBMS. The creation of an authentication page should be another relatively simple addition of one small database table, one new ASP.NET form, and several small additions to the existing forms to ensure authentication and authorization.

Fortunately, the future difficulties that can be foreseen are side effects of planning decisions made with conscious awareness of those potential difficulties. The implementation and testing went fairly smoothly, especially considering that much of the knowledge necessary to complete the project was acquired with the aid of reference books and classes during the implementation process.

Overall, the project is considered a success by the owner of Lightspeed. The database and code generated for this project will likely be utilized in one form or another for many years.

XI. References

Holzner, S. (2003). ***Teach Yourself Microsoft Visual Basic .NET 2003 in 21 Days*****: Sams Publishing.**

Walther, S. (2004). ***ASP.NET Unleashed*** **(Second ed.): Sams Publishing.**

APPENDIX A – VB.NET Code

Validate Module

```
Module Validation
    'Path to Database, can be changed freely
    'note that if type changes, connection strings in all
    'Connection Objects must be changed
    Public strDatabasePath As String = "c:\lightspeed\lightspeed.mdb"

    'Validates Email Addresses via Regular Expression
    Function EmailValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New
System.Text.RegularExpressions.Regex("\w+([-+.]\w+)*@\w+([-
.]\w+)*\.\w+([-.]\w+)*")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If

    End Function
    'Validates ZIP Codes via Regular Expression
    Function ZIPValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[0-
9]{5}(\-[0-9]{4}){0,1}$")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function
    'Validates State Abbreviation via Regular Expression
    Function StateValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[A-
Z]{2}$")
        If regexVal.IsMatch(strTest.Trim.ToUpper) Or strTest.Trim = ""
Then
            Return True
        Else
            Return False
        End If
    End Function
    'Validates Phone Number via Regular Expression
    Function PhoneValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[0-
9]{3}\-[0-9]{3}\-[0-9]{4}$")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function
    'Expansion of IsNumeric, returns false if the string includes a $
    Function TrueIsNumeric(ByVal strTest As String) As Boolean
        If IsNumeric(strTest) And Not strTest.StartsWith("$") Then
```

```
            Return True
        Else
            Return False
        End If
    End Function
    'Simple validation, verifying a string is not blank
    Function RequiredValidation(ByVal strTest As String) As Boolean
        If Not strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function
End Module
```

View Customers Form

```
    'creates connection string, fills datagrid
    Private Sub frmViewCust_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        OleConDB.ConnectionString = "Jet OLEDB:Global Partial Bulk
Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Database Password=;Data Source=""" & strDatabasePath & """;" _
            & "Password=;Jet OLEDB:Engine Type=5;Jet OLEDB:Global Bulk
Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;Extended Properties=;" _
            & "Mode=Share Deny None;Jet OLEDB:New Database
Password=;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=False;User ID=Admin;Jet OLEDB:Encrypt Database=False"
        dstCustomers.Clear()
        OleAdapDB.Fill(dstCustomers)
        dtgCustomers.DataMember = "customers"
    End Sub

    'opens the Update Customer Record form using selected CustID,
disables self
    Private Sub btnEditRecord_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEditRecord.Click
        Dim intRow As Integer
        Dim strQuery As String
        intRow = dtgCustomers.CurrentRowIndex
        strQuery = "select * from customers where CustID=" &
dtgCustomers.Item(intRow, 0)
        'if there is no record selected, error
        If Not intRow = -1 Then
            Dim frmUpAddCust As New frmUpAddCust
            frmUpAddCust.OleDbDataAdapter1.SelectCommand.CommandText =
strQuery
            frmUpAddCust.strType = "Edit"
            frmUpAddCust.frmViewCust = Me
            Me.Enabled = False
            frmUpAddCust.Show()
        Else
            MsgBox("No Row is Selected")
        End If
    End Sub

    'opens the Add Customer Record form, disables self
    Private Sub btnAddRecord_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnAddRecord.Click
        Dim frmUpAddCust As New frmUpAddCust
        frmUpAddCust.strType = "Add"
        frmUpAddCust.frmViewCust = Me
        Me.Enabled = False
        frmUpAddCust.Show()
    End Sub

    'creates query string and fills the datagrid with the results
```

```
    Private Sub btnQuickSearch_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnQuickSearch.Click
        Dim strName As String
        Dim strQuery As String
        strName = txtQuickSearch.Text
        strQuery = "select * from customers where FirstName Like '%" &
strName & "%' or LastName Like '%" & strName & "%'"

        OleAdapDB.SelectCommand.CommandText = strQuery
        dstCustomers.Clear()
        OleAdapDB.Fill(dstCustomers)
        dtgCustomers.DataMember = "customers"
    End Sub

    'opens the View Transactions form, sets Query to select records
match selected CustID
    Private Sub btnViewTransactions_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnViewTransactions.Click
        Dim intRow As Integer
        Dim strQuery As String
        intRow = dtgCustomers.CurrentRowIndex

        'if there is no record selected, error
        If Not intRow = -1 Then
            strQuery = "SELECT customers.CustID, transactions.TransID,
customers.FirstName, customers.LastName, transactions.[Date],
transactions.PaidInFull, " & _
                "transactions.HardwareValue + transactions.LaborValue +
transactions.SoftwareValue + transactions.TaxValue AS TotalValue,
transactions.DownPayment " & _
                "FROM (transactions INNER JOIN customers ON
transactions.CustID = customers.CustID) where customers.CustID=" &
dtgCustomers.Item(intRow, 0)

            Dim frmViewTrans As New frmViewTransactions
            frmViewTrans.frmViewCust = Me
            frmViewTrans.OleAdapDB.SelectCommand.CommandText = strQuery
            frmViewTrans.Show()
            Me.Enabled = False
        Else
            MsgBox("No Row is Selected")
        End If
    End Sub

    'opens an Add Transaction form, sends current selected CustID
    Private Sub btnAddTrans_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnAddTrans.Click
        Dim intRow As Integer
        Dim strQuery As String
        intRow = dtgCustomers.CurrentRowIndex

        'if there is no record selected, error
        If Not intRow = -1 Then
            Dim frmAddTrans As New frmUpAddTrans
            frmAddTrans.strType = "Add"
            frmAddTrans.intCustID = dtgCustomers.Item(intRow, 0)
```

```
            frmAddTrans.frmReferrer = Me
            frmAddTrans.Show()
            Me.Enabled = False
        Else
            MsgBox("No Row is Selected")
        End If
    End Sub

    'shows the Search form, disables the current form
    Private Sub btnSearch_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click
        Dim frmAdvancedSearch As New frmAdvancedSearch
        frmAdvancedSearch.frmViewCust = Me
        Me.Enabled = False
        frmAdvancedSearch.Show()
    End Sub

    'support function for search page, executes query and fills
datagrid
    Public Sub PerformSearch()
        Me.Enabled = True
        dstCustomers.Clear()
        OleAdapDB.Fill(dstCustomers)
        dtgCustomers.DataMember = "customers"
    End Sub
End Class
```

View Transactions Form

```
    Dim strBaseQuery As String
    Public frmViewCust As frmViewCust

    'creates connection string, fills datagrid, sets strBaseQuery
    Private Sub frmViewCust_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
        OleConDB.ConnectionString = "Jet OLEDB:Global Partial Bulk
Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Database Password=;Data Source=""" & strDatabasePath & """;" _
            & "Password=;Jet OLEDB:Engine Type=5;Jet OLEDB:Global Bulk
Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;Extended Properties=;" _
            & "Mode=Share Deny None;Jet OLEDB:New Database
Password=;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=False;User ID=Admin;Jet OLEDB:Encrypt Database=False"

        dstTransactions.Clear()
        OleAdapDB.Fill(dstTransactions)
        dtgTransactions.DataMember = "transactions"

        strBaseQuery = OleAdapDB.SelectCommand.CommandText
    End Sub

    'sends the current selected entry to the Update Form
    Private Sub btnEditRecord_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEditRecord.Click
        Dim intRow As Integer
        Dim strQuery As String
        intRow = dtgTransactions.CurrentRowIndex

        'if the row exists, execute - if not, error
        If Not intRow = -1 Then
            strQuery = "select * from transactions where TransID=" &
dtgTransactions.Item(intRow, 0)
            Dim frmUpTrans As New frmUpAddTrans
            frmUpTrans.OleDbDataAdapter1.SelectCommand.CommandText =
strQuery
            frmUpTrans.strType = "Edit"
            frmUpTrans.frmViewTrans = Me
            frmUpTrans.frmReferrer = Me
            frmUpTrans.Show()
            Me.Enabled = False
        Else
            MsgBox("No Row is Selected")
        End If
    End Sub

    'creates query string and fills the datagrid with the results
    Private Sub btnQuickSearch_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnQuickSearch.Click
        Dim strQuery As String
```

```
        strQuery = strBaseQuery & " AND (transactions.[Date] BETWEEN #"
& CalStartDate.SelectionStart & "# AND #" & calEndDate.SelectionStart &
"#)"
        OleAdapDB.SelectCommand.CommandText = strQuery
        dstTransactions.Clear()
        OleAdapDB.Fill(dstTransactions)
        dtgTransactions.DataMember = "transactions"
    End Sub

    'Cleanup - enables referring form
    Private Sub frmViewTransactions_Closed(ByVal sender As Object,
ByVal e As System.EventArgs) Handles MyBase.Closed
        frmViewCust.Enabled = True
    End Sub
End Class
```

Add/Update Customer Form

```
    Public frmViewCust As frmViewCust
    Public strType As String

    'the following were created by wizard
    Private Sub btnCancel_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnCancel.Click
    Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnUpdate.Click
    Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLoad.Click
    Public Sub UpdateDataSet()
    Public Sub LoadDataSet()
    Public Sub UpdateDataSource(ByVal ChangedRows As
    Public Sub FillDataSet(ByVal dataSet As LightSpeedVB.dstUpAddCust)
    'end auto-generated functions

    'loads the dataset and adds a record, if necessary
    Private Sub frmUpAddCust_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial
Bulk Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database Locking
Mode=1;Jet OLEDB:Database Password=;Data Source=""" & strDatabasePath &
""";" _
            & "Password=;Jet OLEDB:Engine Type=5;Jet OLEDB:Global Bulk
Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;Extended Properties=;" _
            & "Mode=Share Deny None;Jet OLEDB:New Database
Password=;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=False;User ID=Admin;Jet OLEDB:Encrypt Database=False"

        Try
            'Attempt to load the dataset.
            Me.LoadDataSet()
        Catch eLoad As System.Exception
            'Add your error handling code here.
            'Display error message, if any.
            System.Windows.Forms.MessageBox.Show(eLoad.Message)
        End Try

        If strType = "Add" Then
            Try
                'Clear out the current edits
                Me.BindingContext(objdstUpAddCust,
"customers").EndCurrentEdit()
                Me.BindingContext(objdstUpAddCust,
"customers").AddNew()
            Catch eEndEdit As System.Exception
                System.Windows.Forms.MessageBox.Show(eEndEdit.Message)
            End Try
        End If
    End Sub
```

```
    'Validates input with a series of if statments, also outputs errors
if needed
    Function ValidateAddUpCust() As Boolean
        Dim boolErrorFlag As Boolean = False
        Dim strErrors As String = "The Following Errors were Found:" &
ControlChars.NewLine & ControlChars.NewLine

        If Not RequiredValidation(editFirstName.Text) Then
            boolErrorFlag = True
            strErrors += "First Name Must Be Entered" &
ControlChars.NewLine
        End If

        If Not StateValidation(editState.Text) Then
            boolErrorFlag = True
            strErrors += "State is Not Valid, should be in abbreviated
format 'SD'" & ControlChars.NewLine
        End If

        If Not ZIPValidation(editZIP.Text) Then
            boolErrorFlag = True
            strErrors += "ZIP Code is Not Valid, should be in the
format '57042' or '57042-1111'" & ControlChars.NewLine
        End If

        If Not PhoneValidation(editPhone.Text) Then
            boolErrorFlag = True
            strErrors += "Phone Number is Not Valid, should be in the
format '605-555-5555''" & ControlChars.NewLine
        End If

        If Not EmailValidation(editEmail.Text) Then
            boolErrorFlag = True
            strErrors += "E-mail Address is Not Valid" &
ControlChars.NewLine
        End If

        If boolErrorFlag Then
            MsgBox(strErrors)
            Return False
        Else
            Return True
        End If
    End Function

    'clean up - enables referring page
    Private Sub frmUpAddCust_Closed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Closed
        frmViewCust.Enabled = True
        frmViewCust.Focus()
    End Sub
End Class
```

Add/Update Transaction Form

```
    Public strType As String
    Public frmViewTrans As frmViewTransactions
    Public intCustID As Integer
    Public frmReferrer As Object

    'the following functions were created by the Data Form Wizard
    Private Sub btnCancel_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnCancel.Click
    Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnDelete.Click
    Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAdd.Click
    Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLoad.Click
    Private Sub btnCancelAll_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnCancelAll.Click
    Public Sub UpdateDataSet()
    Public Sub LoadDataSet()
    Public Sub UpdateDataSource(ByVal ChangedRows As
LightSpeedVB.dstUpAddTrans)
    Public Sub FillDataSet(ByVal dataSet As LightSpeedVB.dstUpAddTrans)
    'end auto-generated functions

    'loads the dataset, if necessary
    Private Sub frmUpAddTrans_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial
Bulk Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database Locking
Mode=1;Jet OLEDB:Database Password=;Data Source=""" & strDatabasePath &
""";" _
            & "Password=;Jet OLEDB:Engine Type=5;Jet OLEDB:Global Bulk
Transactions=1;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System
database=;Jet OLEDB:SFP=False;Extended Properties=;" _
            & "Mode=Share Deny None;Jet OLEDB:New Database
Password=;Jet OLEDB:Create System Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=False;User ID=Admin;Jet OLEDB:Encrypt Database=False"

        If strType = "Edit" Then
            Try
                'Attempt to load the dataset.
                Me.LoadDataSet()
            Catch eLoad As System.Exception
                'Add your error handling code here.
                'Display error message, if any.
                System.Windows.Forms.MessageBox.Show(eLoad.Message)
            End Try
            Dim datRecordDate As Date
            datRecordDate = editDate.Text
            calEdit.SetDate(datRecordDate)
        End If
    End Sub

    'Inserts or Updates are record, as appropriate
```

```
    Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnUpdate.Click
        If ValidateAddUpTrans() Then
            'handles updates differently depending on type
            If strType = "Edit" Then
                Try
                    'Attempt to update the datasource.
                    Me.UpdateDataSet()
                Catch eUpdate As System.Exception
                    'Add your error handling code here.
                    'Display error message, if any.

System.Windows.Forms.MessageBox.Show(eUpdate.Message)
                End Try

                frmViewTrans.dstTransactions.Clear()

frmViewTrans.OleAdapDB.Fill(frmViewTrans.dstTransactions)
            ElseIf strType = "Add" Then
                Dim conInsert As System.Data.OleDb.OleDbConnection
                Dim strInsert As String
                Dim cmdInsert As System.Data.OleDb.OleDbCommand

                conInsert = New
System.Data.OleDb.OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DAT
A Source=" & strDatabasePath)
                strInsert = "insert into transactions (CustID, [Date],
HardwareValue, SoftwareValue, LaborValue, TaxValue, DownPayment, Notes,
PaidInFull) Values (@CustID, @Date, @HardwareValue, @SoftwareValue,
@LaborValue, @TaxValue, @DownPayment, @Notes, @PaidInFull)"
                cmdInsert = New
System.Data.OleDb.OleDbCommand(strInsert, conInsert)

                cmdInsert.Parameters.Add("@CustID", intCustID)
                cmdInsert.Parameters.Add("@Date",
calEdit.SelectionStart)
                cmdInsert.Parameters.Add("@HardwareValue",
editHardwareValue.Text)
                cmdInsert.Parameters.Add("@SoftwareValue",
editSoftwareValue.Text)
                cmdInsert.Parameters.Add("@LaborValue",
editLaborValue.Text)
                cmdInsert.Parameters.Add("@TaxValue",
editTaxValue.Text)
                cmdInsert.Parameters.Add("@DownPayment",
editDownPayment.Text)
                cmdInsert.Parameters.Add("@Notes", editNotes.Text)
                cmdInsert.Parameters.Add("@PaidInFull",
editPaidInFull.Checked)

                conInsert.Open()
                cmdInsert.ExecuteNonQuery()
                conInsert.Close()
            End If
            Me.Close()
        End If
    End Sub
```

```
    'used to keep the pre-generated textbox consistent with catalog
    'facilitates usage of pre-generate functions
    Private Sub calEdit_DateChanged(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.DateRangeEventArgs) Handles
calEdit.DateChanged
        editDate.Text = calEdit.SelectionStart
    End Sub

    'clean up - enables referring form
    Private Sub frmUpAddTrans_Closed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Closed
        frmReferrer.Enabled = True
        frmReferrer.Focus()
    End Sub

    'validates and outputs any error messages necessary
    Function ValidateAddUpTrans() As Boolean
        Dim boolErrorFlag As Boolean = False
        Dim strErrors As String = "The Following Errors were Found:" &
ControlChars.NewLine & ControlChars.NewLine

        If Not TrueIsNumeric(editHardwareValue.Text) Then
            boolErrorFlag = True
            strErrors += "HardwareValue Must Be Numeric" &
ControlChars.NewLine
        End If

        If Not TrueIsNumeric(editSoftwareValue.Text) Then
            boolErrorFlag = True
            strErrors += "SoftwareValue Must Be Numeric" &
ControlChars.NewLine
        End If

        If Not TrueIsNumeric(editLaborValue.Text) Then
            boolErrorFlag = True
            strErrors += "LaborValue Must Be Numeric" &
ControlChars.NewLine
        End If

        If Not TrueIsNumeric(editTaxValue.Text) Then
            boolErrorFlag = True
            strErrors += "TaxValue Must Be Numeric" &
ControlChars.NewLine
        End If

        If Not TrueIsNumeric(editDownPayment.Text) Then
            boolErrorFlag = True
            strErrors += "DownPayment Must Be Numeric" &
ControlChars.NewLine
        End If

        'if all of the values were numbers, check to make sure the
total isn't zero
        If Not boolErrorFlag Then
            If (editHardwareValue.Text + editSoftwareValue.Text +
editLaborValue.Text + editTaxValue.Text) = 0 Then
```

```
                boolErrorFlag = True
                strErrors += "The Total Value cannot be Zero!  Numeric
Values must be entered."
            End If
        End If

        If boolErrorFlag Then
            MsgBox(strErrors)
            Return False
        Else
            Return True
        End If
    End Function
End Class
```

Advanced Search Form

```
    Public frmViewCust As frmViewCust
    Dim strQuery As String
    Dim strErrors As String
    Dim boolErrorFlag As Boolean = False
    Dim boolSearchSuccessFlag As Boolean = False

    'Query Bulider for Transaction Searches
    Private Sub handleTransactionQuery()
        Dim intCount As Integer = 0
        boolErrorFlag = False
        strErrors = "The Following Errors were Detected:" &
ControlChars.NewLine & ControlChars.NewLine
        strQuery = "select *, customers.CustID, customers.FirstName,
customers.LastName, HardwareValue+SoftwareValue+LaborValue+TaxValue as
TotalValue From (transactions INNER JOIN customers ON
transactions.CustID = customers.CustID)"

        'the following if statements build the query, validate input,
and create error messages
        If chkTransID.Checked Then
            If TrueIsNumeric(txtTransID.Text) Then
                checkFirstEntry(intCount)
                strQuery += " TransID=" & txtTransID.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "TransID Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkCustIDTrans.Checked Then
            If TrueIsNumeric(txtCustIDTrans.Text) Then
                checkFirstEntry(intCount)
                strQuery += " transactions.CustID=" &
txtCustIDTrans.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "CustID Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkNameTrans.Checked Then
            checkFirstEntry(intCount)
            strQuery += " (FirstName Like '%" & txtNameTrans.Text & "%'
or LastName Like '%" & txtNameTrans.Text & "%') "
        End If

        If chkDate.Checked Then
            checkFirstEntry(intCount)
            If cboDate.SelectedItem = "Equal" Then
                strQuery += " [Date] = #" & calStartDate.SelectionStart
& "# "
            Else
```

```
                strQuery += " ([Date] BETWEEN #" &
calStartDate.SelectionStart & "# AND #" & calEndDate.SelectionStart &
"#) "
            End If
        End If

        If chkHardwareValue.Checked Then
            If TrueIsNumeric(txtHardwareValue.Text) Then
                checkFirstEntry(intCount)
                strQuery += " HardwareValue" &
cboHardwareValueType.SelectedItem & txtHardwareValue.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "HardwareValue Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkSoftwareValue.Checked Then
            If TrueIsNumeric(txtSoftwareValue.Text) Then
                checkFirstEntry(intCount)
                strQuery += " SoftwareValue" &
cboSoftwareValueType.SelectedItem & txtSoftwareValue.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "SoftwareValue Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkLaborValue.Checked Then
            If TrueIsNumeric(txtLaborValue.Text) Then
                checkFirstEntry(intCount)
                strQuery += " LaborValue" &
cboLaborValueType.SelectedItem & txtLaborValue.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "LaborValue Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkTaxValue.Checked Then
            If TrueIsNumeric(txtTaxValue.Text) Then
                checkFirstEntry(intCount)
                strQuery += " TaxValue" & cboTaxValueType.SelectedItem
& txtTaxValue.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "TaxValue Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkDownPayment.Checked Then
            If TrueIsNumeric(txtDownPayment.Text) Then
                checkFirstEntry(intCount)
```

```
                strQuery += " DownPayment" &
cboDownPaymentType.SelectedItem & txtDownPayment.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "DownPayment Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkNotesTrans.Checked Then
            checkFirstEntry(intCount)
            strQuery += " transactions.Notes Like '%" &
txtNotesTrans.Text & "%' "
        End If

        If chkPaidInFull.Checked Then
            checkFirstEntry(intCount)
            strQuery += " PaidInFull=" & chkPaidInFullValue.Checked & "
"
        End If
    End Sub

    'Query Builder for Customer Searches
    Private Sub handleCustomerQuery()
        Dim intCount As Integer = 0
        strQuery = "select * From Customers"
        boolErrorFlag = False
        strErrors = "The Following Errors were Detected:" &
ControlChars.NewLine & ControlChars.NewLine

        'the following if statements build the query, validate input,
and create error messages
        If chkCustID.Checked Then
            If TrueIsNumeric(txtCustID.Text) Then
                checkFirstEntry(intCount)
                strQuery += " CustID=" & txtCustID.Text & " "
            Else
                boolErrorFlag = True
                strErrors += "CustID Must Be Numeric" &
ControlChars.NewLine
            End If
        End If

        If chkFirstName.Checked Then
            checkFirstEntry(intCount)
            strQuery += " FirstName Like '%" & txtFirstName.Text & "%'
"
        End If

        If chkLastName.Checked Then
            checkFirstEntry(intCount)
            strQuery += " LastName Like '%" & txtLastName.Text & "%' "
        End If

        If chkFirstLast.Checked Then
            checkFirstEntry(intCount)
```

```
            strQuery += " (FirstName Like '%" & txtFirstLast.Text & "%'
or LastName Like '%" & txtFirstLast.Text & "%')"
        End If

        If chkPhone.Checked Then
            checkFirstEntry(intCount)
            strQuery += " Phone Like '%" & txtPhone.Text & "%' "
        End If

        If chkEmail.Checked Then
            checkFirstEntry(intCount)
            strQuery += " Email Like '%" & txtEmail.Text & "%' "
        End If

        If chkAddress.Checked Then
            checkFirstEntry(intCount)
            strQuery += " Address Like '%" & txtAddress.Text & "%' "
        End If

        If chkCity.Checked Then
            checkFirstEntry(intCount)
            strQuery += " City Like '%" & txtCity.Text & "%' "
        End If

        If chkState.Checked Then
            checkFirstEntry(intCount)
            strQuery += " State Like '%" & txtState.Text & "%' "
        End If

        If chkZIP.Checked Then
            checkFirstEntry(intCount)
            strQuery += " ZIP Like '%" & txtZIP.Text & "%' "
        End If

        If chkNotes.Checked Then
            checkFirstEntry(intCount)
            strQuery += " Notes Like '%" & txtNotes.Text & "%' "
        End If
    End Sub

    'Support function for Query Builders, ensures correct string format
    Private Sub checkFirstEntry(ByRef intCount As Integer)
        If intCount = 0 Then
            strQuery = strQuery + " where "
        Else
            strQuery = strQuery + " and "
        End If
        intCount = intCount + 1
    End Sub

    'toggles the label attached to the checkbox when value is changed
    Private Sub chkPaidInFullValue_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
chkPaidInFullValue.CheckedChanged
        If chkPaidInFullValue.Checked Then
            chkPaidInFullValue.Text = "Paid"
        Else
```

```
            chkPaidInFullValue.Text = "Unpaid"
        End If
    End Sub

    'ensures the correct calander objects are visible for searching
    Private Sub cboDate_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cboDate.SelectedIndexChanged
        If cboDate.SelectedItem = "Equal" Then
            calEndDate.Visible = False
            lblAnd.Visible = False
        Else
            calEndDate.Visible = True
            lblAnd.Visible = True
        End If
    End Sub

    'runs correct query generator, outputs errors, and redirects
    Private Sub btnSearch_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click
        'run the correct query generator based on selected tab
        If tabSearch.SelectedTab.Text = "Customers" Then
            handleCustomerQuery()
        ElseIf tabSearch.SelectedTab.Text = "Transactions" Then
            handleTransactionQuery()
        ElseIf tabSearch.SelectedTab.Text = "Contract Summary" Then
            If ValidateAnalysis() Then
                PerformAnalysis()
            End If
        End If

        'display message if there were errors, send query to proper
form if not
        If boolErrorFlag Then
            MsgBox(strErrors)
        Else
            boolSearchSuccessFlag = True
            If tabSearch.SelectedTab.Text = "Customers" Then
                frmViewCust.OleAdapDB.SelectCommand.CommandText =
strQuery
                frmViewCust.PerformSearch()
                Me.Close()
            ElseIf tabSearch.SelectedTab.Text = "Transactions" Then
                Dim frmViewTrans As New frmViewTransactions
                frmViewTrans.frmViewCust = Me.frmViewCust
                frmViewTrans.OleAdapDB.SelectCommand.CommandText =
strQuery
                frmViewTrans.Show()
                Me.Close()

                'special case - results boolSearchSuccessFlag is search
successful
                'but handled by current page
            ElseIf tabSearch.SelectedTab.Text = "Contract Summary" Then
                boolSearchSuccessFlag = False
            End If
        End If
```

```
    End Sub

    'initialization for combo boxes and calendar objects
    Private Sub frmAdvancedSearch_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        cboHardwareValueType.SelectedItem = "="
        cboSoftwareValueType.SelectedItem = "="
        cboLaborValueType.SelectedItem = "="
        cboTaxValueType.SelectedItem = "="
        cboDownPaymentType.SelectedItem = "="
        cboDate.SelectedItem = "Equal"

calStartDateAnalysis.SetDate(calEndDateAnalysis.SelectionStart.AddYears
(-1))
    End Sub

    'Cleanup - enables referring form
    Private Sub frmAdvancedSearch_Closed(ByVal sender As Object, ByVal
e As System.EventArgs) Handles MyBase.Closed
        If Not boolSearchSuccessFlag Then
            frmViewCust.Enabled = True
        End If
    End Sub

    'loads summary data from database and outputs that data
    Private Sub PerformAnalysis()
        Dim conDB As System.Data.OleDb.OleDbConnection
        Dim cmdSelect As System.Data.OleDb.OleDbCommand
        Dim dtrAnalysis As System.Data.OleDb.OleDbDataReader
        Dim strSelect As String
        conDB = New
System.Data.OleDb.OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DAT
A Source=" & strDatabasePath)

        'if searching by a CustID, load the Customer Names
        If chkCustIDAnalysis.Checked Then
            cmdSelect = New System.Data.OleDb.OleDbCommand("select
FirstName, LastName from Customers where CustID=" &
txtCustIDAnalysis.Text, conDB)
            conDB.Open()
            dtrAnalysis = cmdSelect.ExecuteReader()
            If dtrAnalysis.Read() Then
                lblName.Text = dtrAnalysis(0) & " " & dtrAnalysis(1)
            Else
                MsgBox("No Record Found!")
                lblName.Text = "No Record Found!"
            End If
            dtrAnalysis.Close()
            conDB.Close()

            strSelect = "select count(TransID), sum(HardwareValue),
Sum(SoftwareValue), Sum(LaborValue), Sum(TaxValue) from transactions
where CustID=" & txtCustIDAnalysis.Text _
            & " and ([Date] between #" &
calStartDateAnalysis.SelectionStart & "# and #" &
calEndDateAnalysis.SelectionStart & "#) "
        Else
```

```
            lblName.Text = "All Customers"
            strSelect = "select count(TransID), sum(HardwareValue),
Sum(SoftwareValue), Sum(LaborValue), Sum(TaxValue) from transactions
where" _
            & " ([Date] between #" &
calStartDateAnalysis.SelectionStart & "# and #" &
calEndDateAnalysis.SelectionStart & "#) "
        End If

        cmdSelect = New System.Data.OleDb.OleDbCommand(strSelect,
conDB)
        conDB.Open()
        dtrAnalysis = cmdSelect.ExecuteReader()
        dtrAnalysis.Read()

        lblTotalTrans.Text = dtrAnalysis(0)

        'if dtrAnalysis=0, the other summary info will be DBNull, test
prevents errors
        If Not dtrAnalysis(0) = 0 Then
            lblTotalHardwareValue.Text = dtrAnalysis(1)
            lblTotalSoftwareValue.Text = dtrAnalysis(2)
            lblTotalLaborValue.Text = dtrAnalysis(3)
            lblTotalTaxValue.Text = dtrAnalysis(4)
            lblTotalValue.Text = dtrAnalysis(1) + dtrAnalysis(2) +
dtrAnalysis(3) + dtrAnalysis(4)
        End If

        dtrAnalysis.Close()
        conDB.Close()
    End Sub

    'validation for the Contract Summary search
    Function ValidateAnalysis() As Boolean
        If chkCustIDAnalysis.Checked AndAlso
TrueIsNumeric(txtCustIDAnalysis.Text) Then
            Return True
        ElseIf Not chkCustIDAnalysis.Checked Then
            Return True
        Else
            MsgBox("Customer ID must be entered and must be Numeric")
            Return False
        End If
    End Function

    'used to ensure no conflics when searching, unchecks conflicting
controls
    Private Sub chkFirstLast_CheckedChanged_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
chkFirstLast.CheckedChanged
        If chkFirstLast.Checked Then
            chkFirstName.Checked = False
            chkLastName.Checked = False
        End If
    End Sub
```

```
    'used to ensure no conflics when searching, unchecks conflicting
control
    Private Sub chkFirstName_CheckedChanged_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
chkFirstName.CheckedChanged
        If chkFirstName.Checked Then
            chkFirstLast.Checked = False
        End If
    End Sub

    'used to ensure no conflics when searching, unchecks conflicting
control
    Private Sub chkLastName_CheckedChanged_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
chkLastName.CheckedChanged
        If chkLastName.Checked Then
            chkFirstLast.Checked = False
        End If
    End Sub
End Class
```

APPENDIX B – ASP.NET Code

Validate.vb

```
Imports System
Imports System.Web.UI

Public Class Validate
  Inherits Page

    'Validates Email Addresses via Regular Expression
    Function EmailValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New
System.Text.RegularExpressions.Regex("\w+([-+.]\w+)*@\w+([-
.]\w+)*\.\w+([-.]\w+)*")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If

    End Function

    'Validates ZIP Codes via Regular Expression
    Function ZIPValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[0-
9]{5}(\-[0-9]{4}){0,1}$")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function

    'Validates State Abbreviation via Regular Expression
    Function StateValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[A-
Z]{2}$")
        If regexVal.IsMatch(strTest.Trim.ToUpper) Or strTest.Trim = ""
Then
            Return True
        Else
            Return False
        End If
    End Function

    'Validates Phone Number via Regular Expression
    Function PhoneValidation(ByVal strTest As String) As Boolean
        Dim regexVal As New System.Text.RegularExpressions.Regex("^[0-
9]{3}\-[0-9]{3}\-[0-9]{4}$")
        If regexVal.IsMatch(strTest.Trim) Or strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function
```

```
    'Expansion of IsNumeric, returns false if the string includes a $
    Function TrueIsNumeric(ByVal strTest As String) As Boolean
        If Microsoft.VisualBasic.IsNumeric(strTest) And Not
strTest.StartsWith("$") Then
            Return True
        Else
            Return False
        End If
    End Function

    'Simple validation, verifying a string is not blank
    Function RequiredValidation(ByVal strTest As String) As Boolean
        If Not strTest.Trim = "" Then
            Return True
        Else
            Return False
        End If
    End Function
End Class
```

Index.aspx

```
<%@ Page Language="VB" Inherits="Validate" Src="Validate.vb" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

    'standard Page Load - handles initialization for page
    Sub Page_Load
       dim intCustID as integer
       dim strView as String

      ' Retrieve Form GET data
       intCustID = convert.toint32(Request.QueryString("CustID"))
       strView = Request.QueryString("View")

       'if returning from "Edit Transaction", handle correctly
        if strView="Trans" then
            lblCustID.Text=intCustID
            lblViewMode.Text="Trans"
            pnlViewCust.Visible=False
            pnlViewTrans.Visible=True
        end if

        'loads the correct data for a view mode
        if lblViewMode.Text="Cust" then
            LoadCustView("")
        else if lblViewMode.Text="Trans"
            LoadTransView("")
        else if lblViewMode.Text="CustSearch"
            LoadCustView(lblTempQuery.text)
        else if lblViewMode.Text="TransSearch"
            LoadTransView(lblTempQuery.text)
        end if

        'initializes the Calendars
        if not ispostback
            dim CurDate as New Date
            curDate = System.DateTime.Now.ToShortDateString
            CalStartDate.SelectedDate=curDate
            CalStartDate.VisibleDate=curDate
            CalEndDate.SelectedDate=curDate
            CalEndDate.VisibleDate=curDate
            CalStartDateAnalysis.SelectedDate=curDate.AddYears(-1)
            CalStartDateAnalysis.VisibleDate=curDate.AddYears(-1)
            CalEndDateAnalysis.SelectedDate=curDate
            CalEndDateAnalysis.VisibleDate=curDate
            txtStartDate.Text=curDate.ToShortDateString
            txtEndDate.Text=curDate.ToShortDateString
        end if
    End Sub

    'Loads information from the DB for Transaction Viewing
    Sub LoadTransView(strSearchQuery as String)
       Dim conDB As OleDbConnection
       dim strSelect as string
```

```
        Dim cmdSelect As OleDbCommand
        conDB = New OleDbConnection(
"PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" & lblDatabasePath.Text
)

        if not strSearchQuery=""
             strSelect=strSearchQuery
        else if lblCustID.Text="" then
             strSelect="select customers.CustID, FirstName, LastName,
TransID, [Date], PaidInFull,
HardwareValue+SoftwareValue+LaborValue+TaxValue as TotalValue,
DownPayment, transactions.Notes from (transactions INNER JOIN customers
ON transactions.CustID = customers.CustID)"
        else
             strSelect="select customers.CustID, FirstName, LastName,
TransID, [Date], PaidInFull,
HardwareValue+SoftwareValue+LaborValue+TaxValue as TotalValue,
DownPayment, transactions.Notes from (transactions INNER JOIN customers
ON transactions.CustID = customers.CustID) where customers.CustID=" &
lblCustID.text
        end if

        cmdSelect = New OleDbCommand(strSelect , conDB)
        conDB.Open()
        dgrdTransactions.DataSource = cmdSelect.ExecuteReader()
        dgrdTransactions.DataBind()
        conDB.Close()
    End Sub

    'Loads information from the DB for Customer Viewing
    Sub LoadCustView(strSearchQuery as String)
        Dim conDB As OleDbConnection
        Dim cmdSelect As OleDbCommand
        conDB = New OleDbConnection(
"PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" & lblDatabasePath.Text
)

        'Uses the Search Query, if it existis
        if strSearchQuery = "" then
             cmdSelect = New OleDbCommand( "Select * From customers",
conDB )
        else
             cmdSelect = New OleDbCommand( strSearchQuery, conDB )
        end if

        conDB.Open()
        dgrdCustomers.DataSource = cmdSelect.ExecuteReader()
        dgrdCustomers.DataBind()
        conDB.Close()
    End Sub

    'if an item is selected on the datagrid, highligh and store its key
in a label
    Sub dgrdTransactions_ItemCommand( s As Object, e As
DataGridCommandEventArgs )
        If e.CommandName="Select" Then
          e.Item.BackColor = System.Drawing.Color.LightBlue
```

```
            e.Item.Font.Bold = True
            lblTransID.text=dgrdTransactions.DataKeys(e.Item.ItemIndex)
            lblCustID.text=e.item.cells(2).text
        End If
    End Sub

    'if an item is selected on the datagrid, highligh and store its key
in a label
    Sub dgrdCustomers_ItemCommand( s As Object, e As
DataGridCommandEventArgs )
        If e.CommandName="Select" Then
            e.Item.BackColor = System.Drawing.Color.LightBlue
            e.Item.Font.Bold = True
            lblCustID.text=dgrdCustomers.DataKeys(e.Item.ItemIndex)
        End If
    End Sub

    'if there is a selected item, redirect - otherwise, show error
message
    Sub btnEditCust_Click(sender As Object, e As EventArgs)
        if not lblCustID.Text="" then
            response.redirect("UpAddCust.aspx?Type=Edit&CustID=" &
lblCustID.text)
            lblCustID.text=""
        else
            lblViewCustErrors.text="No Record Selected"
            lblViewCustErrors.visible=true
        end if
    End Sub

    'redirect to AddCust page
    Sub btnAddCust_Click(sender As Object, e As EventArgs)
            response.redirect("UpAddCust.aspx?Type=Add")
            lblCustID.text=""
    End Sub

    'if there is a selected item, redirect - otherwise, show error
message
    Sub btnViewTrans_Click(sender As Object, e As EventArgs)
        if not lblCustID.Text="" then
            pnlViewCust.Visible=False
            pnlViewTrans.Visible=True
            lblViewMode.text="Trans"
            LoadTransView("")
        else
            lblViewCustErrors.text="No Record Selected"
            lblViewCustErrors.visible=true
        end if
    End Sub

    'Returns from TransView to CustView, avoiding cleanup by
redirecting
    Sub btnReturn_Click(sender As Object, e As EventArgs)
        response.redirect("index.aspx")
    End Sub
```

```
    'if there are selected items, redirect - otherwise, show error
message
    Sub btnEditTrans_Click(sender As Object, e As EventArgs)
        if not lblCustID.Text="" and not lblTransID.Text="" then
            response.redirect("UpAddTrans.aspx?Type=Edit&TransID=" &
lblTransID.text & "&CustID=" & lblCustID.Text)
            lblCustID.Text=""
            lblTransID.text=""
        else
            lblViewTransErrors.text="No Record Selected"
            lblViewTransErrors.visible=true
        end if
    End Sub

    'if there is a selected item, redirect - otherwise, show error
message
    Sub btnAddTransaction_Click(sender As Object, e As EventArgs)
        if not lblCustID.Text="" then
            response.redirect("UpAddTrans.aspx?Type=Add&CustID=" &
lblCustID.Text)
            lblCustID.text=""
        else
            lblViewCustErrors.text="No Record Selected"
            lblViewCustErrors.visible=true
        end if
    End Sub

    'sends user to the search panel, goes to last selected panel
    Sub btnSearch_Click(sender As Object, e As EventArgs)
        pnlViewCust.Visible=False
        pnlSearch.Visible=True
        if cboSearchType.SelectedItem.Value="Customers"
            pnlCustomers.Visible=True
        else if cboSearchType.SelectedItem.Value="Transactions"
            pnlTransactions.Visible=True
        else if cboSearchType.SelectedItem.Value="Contract Summary"
            pnlAnalysis.Visible=True
        end if
    End Sub

    'shows the correct panels for the selected item
    Sub cboSearchType_SelectedIndexChanged(sender As Object, e As
EventArgs)
        If cboSearchType.SelectedItem.Value="Customers" Then
            pnlCustomers.Visible=True
            pnlTransactions.Visible=False
            pnlAnalysis.visible=false
        Else if cboSearchType.SelectedItem.Value="Transactions" Then
            pnlTransactions.Visible=True
            pnlCustomers.Visible=False
            pnlAnalysis.visible=false
        Else if cboSearchType.SelectedItem.Value="Contract Summary"
Then
            pnlAnalysis.visible=true
            pnlTransactions.visible=false
            pnlCustomers.visible=false
        End If
```

```
    End Sub

    Sub cboDate_SelectedIndexChanged(sender As Object, e As EventArgs)
        if cboDate.SelectedItem.Value="Between" then
            calEndDate.Visible=True
            lblAnd.Visible=True
        else
            calEndDate.visible=false
            lblAnd.Visible=false
        end if
    End Sub

    'keeps the Calendar object synchronized with an associated textbox
    Sub calStartDate_SelectionChanged(sender As Object, e As EventArgs)
        txtStartDate.Text=calStartDate.SelectedDate
    End Sub

    'keeps the Calendar object synchronized with an associated textbox
    Sub calEndDate_SelectionChanged(sender As Object, e As EventArgs)
        txtEndDate.Text=calEndDate.SelectedDate
    End Sub

    'will select the correct Query type and run it, ensuring necessary
items are visible or invisible
    Sub btnSearchEx_Click(sender As Object, e As EventArgs)
        if cboSearchType.SelectedItem.Value="Customers" then
            if handleCustomerQuery() then
                lblViewMode.text="CustSearch"
                lblTempQuery.text=strQuery
                LoadCustView(strQuery)
                pnlSearch.Visible=False
                pnlViewCust.Visible=True
            end if
        else if cboSearchType.SelectedItem.Value="Transactions" then
            if handleTransactionQuery() then
                lblViewMode.text="TransSearch"
                lblTempQuery.text=strQuery
                LoadTransView(strQuery)
                pnlSearch.Visible=False
                pnlViewTrans.Visible=True
            end if
        else if cboSearchType.SelectedItem.Value="Contract Summary"
            If ValidateAnalysis() Then
                PerformAnalysis()
            End If
        end if
    End Sub

    '<<<<<<<<<<>>>>>>>>>>
    'Imported VB.NET Search Code
    '<<<<<<<<<<>>>>>>>>>>
    Dim strQuery as String

        'function to build the Customer Search queries and handle
errors
        Function handleCustomerQuery() as Boolean
            Dim intCount As Integer = 0
```

```
            Dim boolErrorFlag as Boolean = False
            Dim strErrors as String = "The Following Errors were
Detected:" & "<br>" & "<br>"
            strQuery = "select * From Customers"

            'if statements to perform validation/query creation
            If chkCustID.Checked Then
                If TrueIsNumeric(txtCustID.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " CustID=" & txtCustID.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "CustID Must Be Numeric" & "<br>"
                End If
            End If

            If chkFirstName.Checked Then
                checkFirstEntry(intCount)
                strQuery += " FirstName Like '%" & txtFirstName.Text &
"%' "
            End If

            If chkLastName.Checked Then
                checkFirstEntry(intCount)
                strQuery += " LastName Like '%" & txtLastName.Text &
"%' "
            End If

            If chkFirstLast.Checked Then
                checkFirstEntry(intCount)
                strQuery += " (FirstName Like '%" & txtFirstLast.Text &
"%' or LastName Like '%" & txtFirstLast.Text & "%')"
            End If

            If chkPhone.Checked Then
                checkFirstEntry(intCount)
                strQuery += " Phone Like '%" & txtPhone.Text & "%' "
            End If

            If chkEmail.Checked Then
                checkFirstEntry(intCount)
                strQuery += " Email Like '%" & txtEmail.Text & "%' "
            End If

            If chkAddress.Checked Then
                checkFirstEntry(intCount)
                strQuery += " Address Like '%" & txtAddress.Text & "%'
"
            End If

            If chkCity.Checked Then
                checkFirstEntry(intCount)
                strQuery += " City Like '%" & txtCity.Text & "%' "
            End If

            If chkState.Checked Then
                checkFirstEntry(intCount)
```

```
                strQuery += " State Like '%" & txtState.Text & "%' "
            End If

            If chkZIP.Checked Then
                checkFirstEntry(intCount)
                strQuery += " ZIP Like '%" & txtZIP.Text & "%' "
            End If

            If chkNotes.Checked Then
                checkFirstEntry(intCount)
                strQuery += " Notes Like '%" & txtNotes.Text & "%' "
            End If

            If boolErrorFlag then
                lblCustSearchErrors.text=strErrors
                lblCustSearchErrors.visible=true
                return false
            else
                return true
            end if
        End Function

        'support function for the query builders, chooses to add
"where" or "and" to
        'the SQL string depending upon positioning
        Private Sub checkFirstEntry(ByRef intCount As Integer)
            If intCount = 0 Then
                strQuery = strQuery + " where "
            Else
                strQuery = strQuery + " and "
            End If
            intCount = intCount + 1
        End Sub

        'function to build the Customer Search queries and handle
errors
        Function handleTransactionQuery() as Boolean
            Dim intCount As Integer = 0
            Dim boolErrorFlag as Boolean = False
            Dim strErrors as String = "The Following Errors were
Detected:" & "<br>" & "<br>"

            strQuery = "select *, customers.CustID as CustID,
FirstName, LastName, transactions.Notes as Notes,
HardwareValue+SoftwareValue+LaborValue+TaxValue as TotalValue From
(transactions INNER JOIN customers ON transactions.CustID =
customers.CustID) "

            'if statements to perform validation/query creation
            If chkTransID.Checked Then
                If TrueIsNumeric(txtTransID.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " TransID=" & txtTransID.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "TransID Must Be Numeric" & "<br>"
                End If
```

```
            End If

            If chkCustIDTrans.Checked Then
                If TrueIsNumeric(txtCustIDTrans.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " transactions.CustID=" &
txtCustIDTrans.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "CustID Must Be Numeric" & "<br>"
                End If
            End If

            If chkNameTrans.Checked Then
                checkFirstEntry(intCount)
                strQuery += " (FirstName Like '%" & txtNameTrans.Text &
"%' or LastName Like '%" & txtNameTrans.Text & "%') "
            End If

            If chkDate.Checked Then
                checkFirstEntry(intCount)
                If cboDate.SelectedItem.value = "Equal" Then
                    strQuery += " [Date] = #" & txtStartDate.text & "#
"
                Else
                    strQuery += " ([Date] BETWEEN #" &
txtStartDate.text & "# AND #" & txtEndDate.text & "#) "
                End If
            End If

            If chkHardwareValue.Checked Then
                If TrueIsNumeric(txtHardwareValue.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " HardwareValue" &
cboHardwareValueType.SelectedItem.Value & txtHardwareValue.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "HardwareValue Must Be Numeric" &
"<br>"
                End If
            End If

            If chkSoftwareValue.Checked Then
                If TrueIsNumeric(txtSoftwareValue.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " SoftwareValue" &
cboSoftwareValueType.SelectedItem.Value & txtSoftwareValue.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "SoftwareValue Must Be Numeric" &
"<br>"
                End If
            End If

            If chkLaborValue.Checked Then
                If TrueIsNumeric(txtLaborValue.Text) Then
                    checkFirstEntry(intCount)
```

```
                    strQuery += " LaborValue" &
cboLaborValueType.SelectedItem.Value & txtLaborValue.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "LaborValue Must Be Numeric" & "<br>"
                End If
            End If

            If chkTaxValue.Checked Then
                If TrueIsNumeric(txtTaxValue.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " TaxValue" &
cboTaxValueType.SelectedItem.Value & txtTaxValue.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "TaxValue Must Be Numeric" & "<br>"
                End If
            End If

            If chkDownPayment.Checked Then
                If TrueIsNumeric(txtDownPayment.Text) Then
                    checkFirstEntry(intCount)
                    strQuery += " DownPayment" &
cboDownPaymentType.SelectedItem.Value & txtDownPayment.Text & " "
                Else
                    boolErrorFlag = True
                    strErrors += "DownPayment Must Be Numeric" & "<br>"
                End If
            End If

            If chkNotesTrans.Checked Then
                checkFirstEntry(intCount)
                strQuery += " transactions.Notes Like '%" &
txtNotesTrans.Text & "%' "
            End If

            If chkPaidInFull.Checked Then
                checkFirstEntry(intCount)
                strQuery += " PaidInFull=" & chkPaidInFullValue.Checked
& " "
            End If

            'output errors if there are any, else return true
            If boolErrorFlag then
                lblTransSearchErrors.text=strErrors
                lblTransSearchErrors.visible=true
                return false
            else
                return true
            end if
        End Function


        'loads summary data from database and outputs that data
        Private Sub PerformAnalysis()
            Dim conDB As System.Data.OleDb.OleDbConnection
            Dim cmdSelect As System.Data.OleDb.OleDbCommand
```

```
            Dim dtrAnalysis As System.Data.OleDb.OleDbDataReader
            Dim strSelect As String
            conDB = New
System.Data.OleDb.OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DAT
A Source=" & lblDatabasePath.Text)


            'if searching by a CustID, load the Customer Names
            If chkCustIDAnalysis.Checked Then
                cmdSelect = New System.Data.OleDb.OleDbCommand("select
FirstName, LastName from Customers where CustID=" &
txtCustIDAnalysis.Text, conDB)
                conDB.Open()
                dtrAnalysis = cmdSelect.ExecuteReader()
                If dtrAnalysis.Read() Then
                    lblName.Text = dtrAnalysis(0) & " " &
dtrAnalysis(1)
                Else
                    lblAnalysisErrors.text="No Record Found!"
                    lblAnalysisErrors.Visible=true
                    lblName.Text = "No Record Found!"
                End If
                dtrAnalysis.Close()
                conDB.Close()

                strSelect = "select count(TransID), sum(HardwareValue),
Sum(SoftwareValue), Sum(LaborValue), Sum(TaxValue) from transactions
where CustID=" & txtCustIDAnalysis.Text _
                & " and ([Date] between #" &
calStartDateAnalysis.SelectedDate & "# and #" &
calEndDateAnalysis.SelectedDate & "#) "
            Else
                lblName.Text = "All Customers"
                strSelect = "select count(TransID), sum(HardwareValue),
Sum(SoftwareValue), Sum(LaborValue), Sum(TaxValue) from transactions
where" _
                & " ([Date] between #" &
calStartDateAnalysis.SelectedDate & "# and #" &
calEndDateAnalysis.SelectedDate & "#) "
            End If

            cmdSelect = New System.Data.OleDb.OleDbCommand(strSelect,
conDB)
            conDB.Open()
            dtrAnalysis = cmdSelect.ExecuteReader()
            dtrAnalysis.Read()

            lblTotalTrans.Text = dtrAnalysis(0)

            'if dtrAnalysis=0, the other summary info will be DBNull,
test prevents errors
            If Not dtrAnalysis(0) = 0 Then
                lblTotalHardwareValue.Text = dtrAnalysis(1)
                lblTotalSoftwareValue.Text = dtrAnalysis(2)
                lblTotalLaborValue.Text = dtrAnalysis(3)
                lblTotalTaxValue.Text = dtrAnalysis(4)
```

```
                lblTotalValue.Text = dtrAnalysis(1) + dtrAnalysis(2) +
dtrAnalysis(3) + dtrAnalysis(4)
            End If

            dtrAnalysis.Close()
            conDB.Close()
        End Sub

        'validation for the Contract Summary search
        Function ValidateAnalysis() As Boolean
            If chkCustIDAnalysis.Checked AndAlso
TrueIsNumeric(txtCustIDAnalysis.Text) Then
                Return True
            ElseIf Not chkCustIDAnalysis.Checked Then
                Return True
            Else
                lblAnalysisErrors.Text="Customer ID must be entered and
must be Numeric"
                lblAnalysisErrors.Visible=True
                Return False
            End If
        End Function

</script>
<html>
<head>
    <title>LightSpeed ASP.NET Interface</title>
</head>
<body>
    <form runat="server">
        <asp:Panel id="pnlViewCust" runat="server">
            <p>
                <asp:Label id="lblViewCustErrors" runat="server"
visible="False" forecolor="Red" enableviewstate="False"></asp:Label>
            </p>
            <p>
                <asp:DataGrid id="dgrdCustomers" DataKeyField="CustID"
Runat="Server" EnableViewState="False" AutoGenerateColumns="False"
OnItemCommand="dgrdcustomers_ItemCommand">
                    <Columns>
                        <asp:ButtonColumn Text="Select"
CommandName="Select"></asp:ButtonColumn>
                        <asp:BoundColumn DataField="CustID"
ReadOnly="True" HeaderText="CustID"></asp:BoundColumn>
                        <asp:BoundColumn DataField="FirstName"
ReadOnly="True" HeaderText="FirstName"></asp:BoundColumn>
                        <asp:BoundColumn DataField="LastName"
ReadOnly="True" HeaderText="LastName"></asp:BoundColumn>
                        <asp:BoundColumn DataField="Phone"
HeaderText="Phone"></asp:BoundColumn>
                        <asp:BoundColumn DataField="Notes"
HeaderText="Notes"></asp:BoundColumn>
                    </Columns>
                </asp:DataGrid>
                <br />
```

```
                <asp:Button id="btnEditCust"
onclick="btnEditCust_Click" runat="server" Text="Edit
Record"></asp:Button>
                <asp:Button id="btnAddCust" onclick="btnAddCust_Click"
runat="server" Text="Add Record"></asp:Button>
                <asp:Button id="btnAddTransaction"
onclick="btnAddTransaction_Click" runat="server" Text="Add
Transaction"></asp:Button>
                <asp:Button id="btnViewTrans"
onclick="btnViewTrans_Click" runat="server" Text="View
Transactions"></asp:Button>
            </p>
            <p>
                <asp:Button id="btnSearch" onclick="btnSearch_Click"
runat="server" Text="Search"></asp:Button>
            </p>
        </asp:Panel>
        <asp:Panel id="pnlViewTrans" runat="server" Visible="False">
            <p>
                <asp:Label id="lblViewTransErrors" runat="server"
forecolor="Red"></asp:Label>
            </p>
            <p>
                <asp:DataGrid id="dgrdTransactions"
DataKeyField="TransID" Runat="Server" EnableViewState="False"
AutoGenerateColumns="False"
OnItemCommand="dgrdtransactions_ItemCommand">
                    <Columns>
                        <asp:ButtonColumn Text="Select"
CommandName="Select"></asp:ButtonColumn>
                        <asp:BoundColumn DataField="TransID"
ReadOnly="True" HeaderText="TransID"></asp:BoundColumn>
                        <asp:BoundColumn DataField="CustID"
ReadOnly="True" HeaderText="CustID"></asp:BoundColumn>
                        <asp:BoundColumn DataField="FirstName"
HeaderText="FirstName"></asp:BoundColumn>
                        <asp:BoundColumn DataField="LastName"
HeaderText="LastName"></asp:BoundColumn>
                        <asp:BoundColumn DataField="Date"
HeaderText="Date" DataFormatString="{0:d}"></asp:BoundColumn>
                        <asp:BoundColumn DataField="PaidInFull"
ReadOnly="True" HeaderText="PaidInFull"></asp:BoundColumn>
                        <asp:BoundColumn DataField="TotalValue"
HeaderText="TotalValue"></asp:BoundColumn>
                        <asp:BoundColumn DataField="DownPayment"
ReadOnly="True" HeaderText="DownPayment"></asp:BoundColumn>
                        <asp:BoundColumn DataField="Notes"
HeaderText="Notes"></asp:BoundColumn>
                    </Columns>
                </asp:DataGrid>
            </p>
            <p>
                <asp:Button id="btnEditTrans"
onclick="btnEditTrans_Click" runat="server" Text="Edit
Transaction"></asp:Button>
            </p>
            <p>
```

```
                <asp:Button id="btnReturn" onclick="btnReturn_Click"
runat="server" Text="Return to Customers"></asp:Button>
            </p>
        </asp:Panel>
        <asp:Panel id="pnlSearch" runat="server" Visible="False">
            <p>
                <asp:DropDownList id="cboSearchType" runat="server"
OnSelectedIndexChanged="cboSearchType_SelectedIndexChanged"
AutoPostBack="True">
                    <asp:ListItem Value="Customers"
Selected="True">Customers</asp:ListItem>
                    <asp:ListItem
Value="Transactions">Transactions</asp:ListItem>
                    <asp:ListItem Value="Contract Summary">Contract
Summary</asp:ListItem>
                </asp:DropDownList>
            </p>
            <p></p>
            <p></p>
            <p>
                <asp:Panel id="pnlCustomers" runat="server"
Visible="False" HorizontalAlign="Left">
                    <asp:Label id="lblCustSearchErrors" runat="server"
forecolor="Red" enableviewstate="False"></asp:Label>
                    <p>
                        <asp:CheckBox id="chkCustID"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label11"
runat="server">CustID:</asp:Label>
                        <asp:TextBox id="txtCustID"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkFirstName"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label1" runat="server">First
Name:</asp:Label>
                        <asp:TextBox id="txtFirstName"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkLastName"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label2" runat="server">Last
Name:</asp:Label>
                        <asp:TextBox id="txtLastName"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkFirstLast"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label10" runat="server">First &
Last Name:</asp:Label>
                        <asp:TextBox id="txtFirstLast"
runat="server"></asp:TextBox>
                    </p>
                    <p>
```

```
                        <asp:CheckBox id="chkAddress"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label3"
runat="server">Address:</asp:Label>
                        <asp:TextBox id="txtAddress"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkCity"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label4"
runat="server">City:</asp:Label>
                        <asp:TextBox id="txtCity"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkState"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label5"
runat="server">State:</asp:Label>
                        <asp:TextBox id="txtState"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkZip"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label6"
runat="server">ZIP:</asp:Label>
                        <asp:TextBox id="txtZIP"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkPhone"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label7"
runat="server">Phone:</asp:Label>
                        <asp:TextBox id="txtPhone"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkEmail"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label8" runat="server">E-
Mail:</asp:Label>
                        <asp:TextBox id="txtEmail"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkNotes"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label9"
runat="server">Notes:</asp:Label>
                        <br />
                        <asp:TextBox id="txtNotes" runat="server"
TextMode="MultiLine" Width="240px" Height="86px"></asp:TextBox>
                    </p>
                </asp:Panel>
```

```
                <asp:Panel id="pnlTransactions" runat="server"
Visible="False" Width="572px">
                    <p>
                        <asp:Label id="lblTransSearchErrors"
runat="server" visible="False" forecolor="Red"
enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:CheckBox id="chkCustIDTrans"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label20"
runat="server">CustID:</asp:Label>
                        <asp:TextBox id="txtCustIDTrans"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkTransID"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label21"
runat="server">TransID:</asp:Label>
                        <asp:TextBox id="txtTransID"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkNameTrans"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label22"
runat="server">Name:</asp:Label>
                        <asp:TextBox id="txtNameTrans"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkDate"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label12"
runat="server">Date:</asp:Label>
                        <asp:DropDownList id="cboDate" runat="server"
OnSelectedIndexChanged="cboDate_SelectedIndexChanged"
AutoPostBack="True">
                            <asp:ListItem Value="Equal"
Selected="True">Equal</asp:ListItem>
                            <asp:ListItem
Value="Between">Between</asp:ListItem>
                        </asp:DropDownList>
                    </p>
                    <p>
                        <asp:Calendar id="calStartDate" runat="server"
OnSelectionChanged="calStartDate_SelectionChanged"></asp:Calendar>
                        <asp:Label id="lblAnd" runat="server"
visible="False" font-bold="True">and</asp:Label>
                        <asp:Calendar id="calEndDate" runat="server"
Visible="False"
OnSelectionChanged="calEndDate_SelectionChanged"></asp:Calendar>
                    </p>
                    <p>
                        <asp:TextBox id="txtStartDate" runat="server"
visible="false"></asp:TextBox>
```

```
                        <asp:TextBox id="txtEndDate" runat="server"
visible="false"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkHardwareValue"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label23"
runat="server">HardwareValue:</asp:Label>
                        <asp:DropDownList id="cboHardwareValueType"
runat="server">
                            <asp:ListItem Value="="
Selected="True">=</asp:ListItem>
                            <asp:ListItem
Value="&gt;">&gt;</asp:ListItem>
                            <asp:ListItem
Value="&lt;">&lt;</asp:ListItem>
                            <asp:ListItem
Value="&gt;=">&gt;=</asp:ListItem>
                            <asp:ListItem
Value="&lt;=">&lt;=</asp:ListItem>
                        </asp:DropDownList>
                        <asp:TextBox id="txtHardwareValue"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkSoftwareValue"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label13"
runat="server">SoftwareValue:</asp:Label>
                        <asp:DropDownList id="cboSoftwareValueType"
runat="server">
                            <asp:ListItem Value="="
Selected="True">=</asp:ListItem>
                            <asp:ListItem
Value="&gt;">&gt;</asp:ListItem>
                            <asp:ListItem
Value="&lt;">&lt;</asp:ListItem>
                            <asp:ListItem
Value="&gt;=">&gt;=</asp:ListItem>
                            <asp:ListItem
Value="&lt;=">&lt;=</asp:ListItem>
                        </asp:DropDownList>
                        <asp:TextBox id="txtSoftwareValue"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkLaborValue"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label15"
runat="server">LaborValue:</asp:Label>
                        <asp:DropDownList id="cboLaborValueType"
runat="server">
                            <asp:ListItem Value="="
Selected="True">=</asp:ListItem>
                            <asp:ListItem
Value="&gt;">&gt;</asp:ListItem>
```

```
                                <asp:ListItem
Value="&lt;">&lt;</asp:ListItem>
                                <asp:ListItem
Value="&gt;=">&gt;=</asp:ListItem>
                                <asp:ListItem
Value="&lt;=">&lt;=</asp:ListItem>
                            </asp:DropDownList>
                            <asp:TextBox id="txtLaborValue"
runat="server"></asp:TextBox>
                        </p>
                        <p>
                            <asp:CheckBox id="chkTaxValue"
runat="server"></asp:CheckBox>
                            <asp:Label id="Label16"
runat="server">TaxValue:</asp:Label>
                            <asp:DropDownList id="cboTaxValueType"
runat="server">
                                <asp:ListItem Value="="
Selected="True">=</asp:ListItem>
                                <asp:ListItem
Value="&gt;">&gt;</asp:ListItem>
                                <asp:ListItem
Value="&lt;">&lt;</asp:ListItem>
                                <asp:ListItem
Value="&gt;=">&gt;=</asp:ListItem>
                                <asp:ListItem
Value="&lt;=">&lt;=</asp:ListItem>
                            </asp:DropDownList>
                            <asp:TextBox id="txtTaxValue"
runat="server"></asp:TextBox>
                        </p>
                        <p>
                            <asp:CheckBox id="chkDownPayment"
runat="server"></asp:CheckBox>
                            <asp:Label id="Label14"
runat="server">DownPayment:</asp:Label>
                            <asp:DropDownList id="cboDownPaymentType"
runat="server">
                                <asp:ListItem Value="="
Selected="True">=</asp:ListItem>
                                <asp:ListItem
Value="&gt;">&gt;</asp:ListItem>
                                <asp:ListItem
Value="&lt;">&lt;</asp:ListItem>
                                <asp:ListItem
Value="&gt;=">&gt;=</asp:ListItem>
                                <asp:ListItem
Value="&lt;=">&lt;=</asp:ListItem>
                            </asp:DropDownList>
                            <asp:TextBox id="txtDownPayment"
runat="server"></asp:TextBox>
                        </p>
                        <p>
                            <asp:CheckBox id="chkPaidInFull"
runat="server"></asp:CheckBox>
                            <asp:Label id="Label18"
runat="server">PaidInFull:</asp:Label>
```

```
                        <asp:TextBox id="txtPaidInFull" runat="server"
Visible="False"></asp:TextBox>
                        <asp:CheckBox id="chkPaidInFullValue"
runat="server"></asp:CheckBox>
                    </p>
                    <p>
                        <asp:CheckBox id="chkNotesTrans"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label19"
runat="server">Notes:</asp:Label>
                        <br />
                        <asp:TextBox id="txtNotesTrans" runat="server"
TextMode="MultiLine" Width="240px" Height="86px"></asp:TextBox>
                    </p>
                </asp:Panel>
            </p>

            <p>
                <asp:Panel id="pnlAnalysis" runat="server"
Visible="False" Width="540px" Height="77px">
                    <p>
                        <asp:Label id="lblAnalysisErrors"
runat="server" forecolor="Red" enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:CheckBox id="chkCustIDAnalysis"
runat="server"></asp:CheckBox>
                        <asp:Label id="Label25"
runat="server">CustID:</asp:Label>
                        <asp:TextBox id="txtCustIDAnalysis"
runat="server"></asp:TextBox>
                    </p>
                    <p>
                        <asp:Label id="Label24"
runat="server">Date:</asp:Label>
                    </p>
                    <p>
                        <asp:Calendar id="calStartDateAnalysis"
runat="server"></asp:Calendar>
                        <asp:Label id="Label17" runat="server" font-
bold="True">and</asp:Label>
                        <asp:Calendar id="calEndDateAnalysis"
runat="server"></asp:Calendar>
                    </p>
                    <p>
                        <asp:Label id="Label26" runat="server" font-
bold="True">Results:</asp:Label>
                    </p>
                    <hr />
                    <p>
                        <asp:Label id="Label28" runat="server" font-
bold="True">Customer: </asp:Label><asp:Label id="lblName"
runat="server" enableviewstate="False" font-bold="True"></asp:Label>
                    </p>
                    <p>
                         <asp:Label id="Label29"
runat="server">Total Hardware Value: </asp:Label><asp:Label
```

```
id="lblTotalHardwareValue" runat="server"
enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:Label id="Label33" runat="server">Total
Software Value: </asp:Label><asp:Label id="lblTotalSoftwareValue"
runat="server" enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:Label id="Label31" runat="server">Total
Labor Value: </asp:Label><asp:Label id="lblTotalLaborValue"
runat="server" enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:Label id="Label36" runat="server">Total
Tax Value: </asp:Label><asp:Label id="lblTotalTaxValue" runat="server"
enableviewstate="False"></asp:Label>
                    </p>
                    <p>
                        <asp:Label id="Label38" runat="server" font-
bold="True">Total Value: </asp:Label><asp:Label id="lblTotalValue"
runat="server" enableviewstate="False" font-bold="True"></asp:Label>
                    </p>
                    <p>
                        <asp:Label id="Label40" runat="server" font-
bold="True">Total Transactions: </asp:Label><asp:Label
id="lblTotalTrans" runat="server" enableviewstate="False" font-
bold="True"></asp:Label>
                    </p>
                </asp:Panel>
            </p>

            <p>
                <asp:Button id="btnSearchEx"
onclick="btnSearchEx_Click" runat="server" Text="Execute
Search"></asp:Button>
            </p>
        </asp:Panel>
        <p>
            <asp:Label id="lblCustID" runat="server"
visible="False"></asp:Label>
        </p>
        <p>
            <asp:Label id="lblTransID" runat="server"
visible="False"></asp:Label>
        </p>
        <p>
            <asp:Label id="lblViewMode" runat="server"
visible="False">Cust</asp:Label>
        </p>
        <p>
            <asp:Label id="lblTempQuery" runat="server"
visible="False"></asp:Label>
        </p>
        <p>
```

```
            <asp:Label id="lblDatabasePath" runat="server"
visible="False"
enabled="False">c:\lightspeed\lightspeed.mdb</asp:Label>
        </p>
    </form>
</body>
</html>
```

UpAddCust.aspx

```
<%@ Page Language="VB" Inherits="Validate" Src="Validate.vb" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

    Dim strType as String
    Dim intCustID as Integer

    Sub Page_Load
       ' Retrieve Form GET data
       intCustID = convert.toint32(Request.QueryString("CustID"))
       strType = Request.QueryString("Type")

        if Not IsPostBack Then
          If strType="Edit" then
           Dim conDB As OleDbConnection
           Dim cmdSelect As OleDbCommand
           Dim dtrRecord as OleDBDataReader

           conDB = New OleDbConnection(
"PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" & lblDatabasePath.text
)
           cmdSelect = New OleDbCommand( "Select * From customers where
CustID=@CustID", conDB )
           cmdSelect.Parameters.Add( "@CustID", intCustID )
           conDB.Open()
           dtrRecord = cmdSelect.ExecuteReader()
           dtrRecord.read()
           txtFirstName.text=dtrRecord("FirstName")
           txtLastName.text=dtrRecord("LastName")
           txtAddress.text=dtrRecord("Address")
           txtCity.text=dtrRecord("City")
           txtState.text=dtrRecord("State")
           txtZIP.text=dtrRecord("ZIP")
           txtPhone.text=dtrRecord("Phone")
           txtEmail.text=dtrRecord("Email")
           txtNotes.text=dtrRecord("Notes")
           conDB.Close()
          End If
        End if
    End Sub

    'Performs the appropriate OLEDB Actions and redirects if Validated
    Sub btnSubmit_Click(sender As Object, e As EventArgs)
        if ValidateCust() then
            if strType="Edit" then
                EditRecord()
            else if strType="Add" then
                AddRecord()
            end if
            response.redirect("index.aspx")
        end if
    End Sub

    'Redirects to Initial Page
```

```
    Sub btnCancel_Click(sender As Object, e As EventArgs)
        response.redirect("index.aspx")
    End Sub

    'OLEDB Commands Necessary to add a New Record
    Sub AddRecord()
       Dim conDB As OleDbConnection
       Dim strInsert As String
       Dim cmdInsert As OleDbCommand

       conDB = New
OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" &
lblDatabasePath.text )
       strInsert = "insert into customers (FirstName, LastName,
Address, City, State, ZIP, Phone, Email, Notes) Values (@FirstName,
@LastName, @Address, @City, @State, @ZIP, @Phone, @Email, @Notes)"
       cmdInsert = New OleDbCommand(strInsert, conDB)

       cmdInsert.Parameters.Add("@FirstName", txtFirstName.Text)
       cmdInsert.Parameters.Add("@LastName", txtLastName.Text)
       cmdInsert.Parameters.Add("@Address", txtAddress.Text)
       cmdInsert.Parameters.Add("@City", txtCity.Text)
       cmdInsert.Parameters.Add("@State", txtState.Text)
       cmdInsert.Parameters.Add("@ZIP", txtZIP.Text)
       cmdInsert.Parameters.Add("@Phone", txtPhone.Text)
       cmdInsert.Parameters.Add("@Email", txtEmail.Text)
       cmdInsert.Parameters.Add("@Notes", txtNotes.Text)

       conDB.Open()
       cmdInsert.ExecuteNonQuery()
       conDB.Close()
    End Sub

    'OLEDB commands necessary to edit an existing record
    Sub EditRecord()
       Dim conDB As OleDbConnection
       Dim strUpdate As String
       Dim cmdUpdate As OleDbCommand

       conDB = New
OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" &
lblDatabasePath.text )
       strUpdate = "Update customers Set FirstName=@FirstName,
LastName=@LastName, Address=@Address, City=@City, State=@State,
ZIP=@ZIP, Phone=@Phone, Email=@Email, Notes=@Notes where
CustID=@CustID"
       cmdUpdate = New OleDbCommand(strUpdate, conDB)

       cmdUpdate.Parameters.Add("@FirstName", txtFirstName.Text)
       cmdUpdate.Parameters.Add("@LastName", txtLastName.Text)
       cmdUpdate.Parameters.Add("@Address", txtAddress.Text)
       cmdUpdate.Parameters.Add("@City", txtCity.Text)
       cmdUpdate.Parameters.Add("@State", txtState.Text)
       cmdUpdate.Parameters.Add("@ZIP", txtZIP.Text)
       cmdUpdate.Parameters.Add("@Phone", txtPhone.Text)
       cmdUpdate.Parameters.Add("@Email", txtEmail.Text)
       cmdUpdate.Parameters.Add("@Notes", txtNotes.Text)
```

```
        cmdUpdate.Parameters.Add("@CustID", intCustID)

        conDB.Open()
        cmdUpdate.ExecuteNonQuery()
        conDB.Close()
    End Sub

    'Validate text - returns true if passed, returns false and outputs
errors if failed
    Function ValidateCust() as Boolean
            Dim boolErrorFlag As Boolean = False
            Dim strErrors As String = "The Following Errors were
Found:" & "<br>" & "<br>"

            If Not RequiredValidation(txtFirstName.Text) Then
                boolErrorFlag = True
                strErrors += "First Name Must Be Entered" & "<br>"
            End If

            If Not StateValidation(txtState.Text) Then
                boolErrorFlag = True
                strErrors += "State is Not Valid, should be in
abbreviated format 'SD'" & "<br>"
            End If

            If Not ZIPValidation(txtZIP.Text) Then
                boolErrorFlag = True
                strErrors += "ZIP Code is Not Valid, should be in the
format '57042' or '57042-1111'" & "<br>"
            End If

            If Not PhoneValidation(txtPhone.Text) Then
                boolErrorFlag = True
                strErrors += "Phone Number is Not Valid, should be in
the format '605-555-5555''" & "<br>"
            End If

            If Not EmailValidation(txtEmail.Text) Then
                boolErrorFlag = True
                strErrors += "E-mail Address is Not Valid" & "<br>"
            End If

            If boolErrorFlag Then
                lblErrors.text=strErrors
                lblErrors.visible=True
                Return False
            Else
                Return True
            End If
    End Function

</script>
<html>
<head>
    <title>LightSpeed ASP.NET - Add/Update Customer</title>
</head>
<body>
```

```
    <form runat="server">
        <p>
            <asp:Label id="lblErrors" runat="server" forecolor="Red"
visible="False"></asp:Label>
        </p>
        <p>
            <asp:Label id="Label1" runat="server">First
Name:</asp:Label>
            <asp:TextBox id="txtFirstName" runat="server"
MaxLength="50"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label2" runat="server">Last
Name:</asp:Label>
            <asp:TextBox id="txtLastName" runat="server"
MaxLength="50"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label3" runat="server">Address:</asp:Label>
            <asp:TextBox id="txtAddress" runat="server"
MaxLength="50"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label4" runat="server">City:</asp:Label>
            <asp:TextBox id="txtCity" runat="server"
MaxLength="50"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label5" runat="server">State:</asp:Label>
            <asp:TextBox id="txtState" runat="server"
MaxLength="2"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label6" runat="server">ZIP:</asp:Label>
            <asp:TextBox id="txtZIP" runat="server"
MaxLength="10"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label7" runat="server">Phone:</asp:Label>
            <asp:TextBox id="txtPhone" runat="server"
MaxLength="14"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label8" runat="server">E-Mail:</asp:Label>
            <asp:TextBox id="txtEmail" runat="server"
MaxLength="50"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label9" runat="server">Notes:</asp:Label>
            <br />
            <asp:TextBox id="txtNotes" runat="server" MaxLength="255"
TextMode="MultiLine" Height="86px" Width="240px"></asp:TextBox>
        </p>
        <p>
            <asp:Button id="btnSubmit" onclick="btnSubmit_Click"
runat="server" Text="Submit"></asp:Button>
```

```
            <asp:Button id="btnCancel" onclick="btnCancel_Click"
runat="server" Text="Cancel"></asp:Button>
        </p>
        <p>
            <asp:Label id="lblDatabasePath" runat="server"
visible="False"
enabled="False">c:\lightspeed\lightspeed.mdb</asp:Label>
        </p>
    </form>
</body>
</html>
```

UpAddTrans.aspx

```
 <%@ Page Language="VB" Inherits="Validate" Src="Validate.vb" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

    Dim strType as String
    Dim intTransID as Integer
    Dim intCustID as Integer

    Sub Page_Load
       ' Retrieve Form GET data
       intTransID = convert.toint32(Request.QueryString("TransID"))
       intCustID = convert.toint32(Request.QueryString("CustID"))
       strType = Request.QueryString("Type")

        'if it's the first page load
        if Not IsPostBack Then
          calDate.SelectedDate=System.DateTime.Now.ToShortDateString
          calDate.VisibleDate=System.DateTime.Now.ToShortDateString
          txtDate.text=calDate.SelectedDate
          'if it's an edit, load the init data from DB
          If strType="Edit" then
           Dim conDB As OleDbConnection
           Dim cmdSelect As OleDbCommand
           Dim dtrRecord as OleDBDataReader
           conDB = New OleDbConnection(
"PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" & lblDatabasePath.Text
)
           cmdSelect = New OleDbCommand( "Select * From transactions
where TransID=@TransID", conDB )
           cmdSelect.Parameters.Add( "@TransID", intTransID )
           conDB.Open()
           dtrRecord = cmdSelect.ExecuteReader()
           dtrRecord.read()
           txtDate.text=dtrRecord("Date")
           txtHardwareValue.text=dtrRecord("HardwareValue")
           txtSoftwareValue.text=dtrRecord("SoftwareValue")
           txtLaborValue.text=dtrRecord("LaborValue")
           txtTaxValue.text=dtrRecord("TaxValue")
           txtDownPayment.text=dtrRecord("DownPayment")
           chkPaidInFull.Checked=dtrRecord("PaidInFull")
           txtNotes.text=dtrRecord("Notes")
           conDB.Close()
           calDate.SelectedDate=txtDate.text
           calDate.VisibleDate=txtDate.text
          End If
        End if
    End Sub

    'performs correct validation/execution depending on type, redirects
appropriately
    Sub btnSubmit_Click(sender As Object, e As EventArgs)
        if ValidateTrans() then
            if strType="Edit" then
                EditRecord()
```

```
response.redirect("index.aspx?View=Trans&CustID="&intCustID)
            else if strType="Add" then
                AddRecord()
                response.redirect("index.aspx")
            end if
        end if
    End Sub

    'procedure to submit changes to DB
    Sub EditRecord()
       Dim conDB As OleDbConnection
       Dim strUpdate As String
       Dim cmdUpdate As OleDbCommand

       conDB = New
OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" &
lblDatabasePath.Text)
       strUpdate = "Update transactions Set [Date]=@Date,
HardwareValue=@HardwareValue, SoftwareValue=@SoftwareValue,
LaborValue=@LaborValue, TaxValue=@TaxValue, DownPayment=@DownPayment,
PaidInFull=@PaidInFull, Notes=@Notes where TransID=@TransID"

       cmdUpdate = New OleDbCommand(strUpdate, conDB)

       cmdUpdate.Parameters.Add("@Date", txtDate.Text)
       cmdUpdate.Parameters.Add("@HardwareValue",
txtHardwareValue.Text)
       cmdUpdate.Parameters.Add("@SoftwareValue",
txtSoftwareValue.Text)
       cmdUpdate.Parameters.Add("@LaborValue", txtLaborValue.Text)
       cmdUpdate.Parameters.Add("@TaxValue", txtTaxValue.Text)
       cmdUpdate.Parameters.Add("@DownPayment", txtDownPayment.Text)
       cmdUpdate.Parameters.Add("@PaidInFull", chkPaidInFull.Checked)
       cmdUpdate.Parameters.Add("@Notes", txtNotes.Text)
       cmdUpdate.Parameters.Add("@TransID", intTransID)

       conDB.Open()
       cmdUpdate.ExecuteNonQuery()
       conDB.Close()
    End Sub

    'procedure to add records to DB
    Sub AddRecord()
       Dim conDB As OleDbConnection
       Dim strInsert As String
       Dim cmdInsert As OleDbCommand

       conDB = New
OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA Source=" &
lblDatabasePath.Text)
       strInsert = "insert into transactions (CustID, [Date],
HardwareValue, SoftwareValue, LaborValue, TaxValue, DownPayment,
PaidInFull, Notes)  Values (@CustID, @Date, @HardwareValue,
@SoftwareValue, @LaborValue, @TaxValue, @DownPayment, @PaidInFull,
@Notes)"
```

```
        cmdInsert = New OleDbCommand(strInsert, conDB)

        cmdInsert.Parameters.Add("@CustID", intCustID)

        cmdInsert.Parameters.Add("@Date", txtDate.Text)
        cmdInsert.Parameters.Add("@HardwareValue",
txtHardwareValue.Text)
        cmdInsert.Parameters.Add("@SoftwareValue",
txtSoftwareValue.Text)
        cmdInsert.Parameters.Add("@LaborValue", txtLaborValue.Text)
        cmdInsert.Parameters.Add("@TaxValue", txtTaxValue.Text)
        cmdInsert.Parameters.Add("@DownPayment", txtDownPayment.Text)
        cmdInsert.Parameters.Add("@PaidInFull", chkPaidInFull.Checked)
        cmdInsert.Parameters.Add("@Notes", txtNotes.Text)

        conDB.Open()
        cmdInsert.ExecuteNonQuery()
        conDB.Close()
    End Sub

    'Validate text - returns true if passed, returns false and outputs
errors if failed
    Function ValidateTrans() As Boolean
        Dim boolErrorFlag As Boolean = False
        Dim strErrors As String = "The Following Errors were Found:" &
"<br>" & "<br>"

        If Not TrueIsNumeric(txtHardwareValue.Text) Then
            boolErrorFlag = True
            strErrors += "HardwareValue Must Be Numeric" & "<br>"
        End If

        If Not TrueIsNumeric(txtSoftwareValue.Text) Then
            boolErrorFlag = True
            strErrors += "SoftwareValue Must Be Numeric" & "<br>"
        End If

        If Not TrueIsNumeric(txtLaborValue.Text) Then
            boolErrorFlag = True
            strErrors += "LaborValue Must Be Numeric" & "<br>"
        End If

        If Not TrueIsNumeric(txtTaxValue.Text) Then
            boolErrorFlag = True
            strErrors += "TaxValue Must Be Numeric" & "<br>"
        End If

        If Not TrueIsNumeric(txtDownPayment.Text) Then
            boolErrorFlag = True
            strErrors += "DownPayment Must Be Numeric" & "<br>"
        End If

       'if all of the values were numbers, check to make sure the total
isn't zero
        If Not boolErrorFlag Then
            If (txtHardwareValue.Text + txtSoftwareValue.Text +
txtLaborValue.Text + txtTaxValue.Text) = 0 Then
```

```
                boolErrorFlag = True
                strErrors += "The Total Value cannot be Zero!  Numeric
Values must be entered."
            End If
        End If

        If boolErrorFlag Then
            lblErrors.text=strErrors
            lblErrors.visible=true
            Return False
        Else
            Return True
        End If
    End Function

    'keeps txtDate and calDate synchronized
    Sub calDate_SelectionChanged(sender As Object, e As EventArgs)
        txtDate.text=calDate.SelectedDate
    End Sub

    'leaves the page, back to referring page
    Sub btnCancel_Click(sender As Object, e As EventArgs)
            if strType="Edit" then

response.redirect("index.aspx?View=Trans&CustID="&intCustID)
            else if strType="Add" then
                response.redirect("index.aspx")
            end if
    End Sub

</script>
<html>
<head>
    <title>LightSpeed ASP.NET - Add/Update Transaction</title>
</head>
<body>
    <form runat="server">
        <p>
            <asp:Label id="lblErrors" runat="server" visible="False"
forecolor="Red"></asp:Label>
        </p>
        <p>
            <asp:Label id="Label1" runat="server">Date:</asp:Label>
            <asp:TextBox id="txtDate" runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Calendar id="calDate" runat="server"
OnSelectionChanged="calDate_SelectionChanged"></asp:Calendar>
        </p>
        <p>
            <asp:Label id="Label2"
runat="server">HardwareValue:</asp:Label>
            <asp:TextBox id="txtHardwareValue"
runat="server"></asp:TextBox>
        </p>
        <p>
```

```
            <asp:Label id="Label3"
runat="server">SoftwareValue:</asp:Label>
            <asp:TextBox id="txtSoftwareValue"
runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label4"
runat="server">LaborValue:</asp:Label>
            <asp:TextBox id="txtLaborValue"
runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label5" runat="server">TaxValue:</asp:Label>
            <asp:TextBox id="txtTaxValue" runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label6"
runat="server">DownPayment:</asp:Label>
            <asp:TextBox id="txtDownPayment"
runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Label id="Label7"
runat="server">PaidInFull:</asp:Label>
            <asp:TextBox id="txtPaidInFull" runat="server"
Visible="False"></asp:TextBox>
            <asp:CheckBox id="chkPaidInFull"
runat="server"></asp:CheckBox>
        </p>
        <p>
            <asp:Label id="Label9" runat="server">Notes:</asp:Label>
            <br />
            <asp:TextBox id="txtNotes" runat="server"
TextMode="MultiLine" Height="86px" Width="240px"
MaxLength="255"></asp:TextBox>
        </p>
        <p>
            <asp:Button id="btnSubmit" onclick="btnSubmit_Click"
runat="server" Text="Submit"></asp:Button>
            <asp:Button id="btnCancel" onclick="btnCancel_Click"
runat="server" Text="Cancel"></asp:Button>
        </p>
        <p>
            <asp:Label id="lblDatabasePath" runat="server"
visible="False"
enabled="False">c:\lightspeed\lightspeed.mdb</asp:Label>
        </p>
    </form>
</body>
</html>
```

APPENDIX C – Project Planning Documentation

Abstract

This project is to create a customer information database with transactional information, primarily to facilitate tracking of contractual work. This database will interact with two different front-ends to simplify its usage and to provide additional functionality. One front end will be an executable written in Visual Basic.NET which will allow a user to add, view, or edit any transaction or customer information. This front end will also allow a user to query the database for any desired information. The other front end will be an ASP.NET web interface which will allow similar functionality in an internet-based application.

Introduction

This project is to create a customer information database with transactional information, primarily to facilitate tracking of contractual work. This database will interact with two different front-ends to simplify its usage and to provide additional functionality.

The client is a small local computer repair shop. In addition to typical PC repair, the shop sells computer/networking hardware and provides network support services. This full-service nature has led to the establishment of several service contracts, but there is currently no method in place to perform any kind of analysis on the contracts. In addition, the growing customer base has brought the need for some kind of customer information database to the forefront.

Statement of problem and summary of current situation

The main problem is that the client has no way to easily track transactional information. There is currently no point-of-sale system in place, and as such there is no electronic record of transactions. The client also has no customer database more sophisticated then an address book, and many customers' information is not even in that resource. However, paper records of many transactions and customers do exist.

The client only has one computer, a standard Windows XP machine with Internet connectivity. There is no server of any kind. This computer is well secured however, running Windows XP Service Pack 2 with a software firewall and protected by a hardware firewall/router which provides network address translation.

Goals

The main goal is to provide a well designed database to store customer and transaction information. The database is the single most important factor in the success of this project, addressing most of the problems directly. The intent is to fully populate the database with all existing customer information and perhaps some existing transactional data. However, to be fully successful, the database must be easily usable.

To make the database more usable, a front-end will be developed in Visual Basic .NET. This interface will provide the ability to insert, edit, and view records stored in the database. Both transactional information and customer information will be input and edited through this form, so it must be capable of handling both types of record. In addition to those functions, it will provide advanced querying features allowing the user to search both customer and transaction information. The results of these searches will also include summary information such as the total value of all listed transactions and the number of transaction listed. This Visual Basic interface will also be designed to allow for relatively easy migration to a different Database Management System (DBMS) in the future. The client considers it a strong possibility that both hardware and software availability will change in the near future, and the Visual Basic interface must be capable of adapting to those possible changes.

In addition to the Visual Basic.NET interface, a web interface in ASP.NET will be created utilizing the same database structure. This interface will not be active at the completion of the project, but the client wishes to have web functionality available should hardware or software upgrades make such functionality practical. In the current situation implementation of a web server and a more secure database is not practical, however.

Like the Visual Basic interface, this code should be as open to future changes as possible, and is intended to allow for easy migration to a different DBMS in the future. While this web interface is not going to be active, the goal is for it to provide the same functionality as the Visual Basic interface. The graphical user interface itself could be very different, but the functionality is to remain virtually unchanged. Less query functionality in the web interface is acceptable to the client because most queries will be run in the office on the Visual Basic interface.

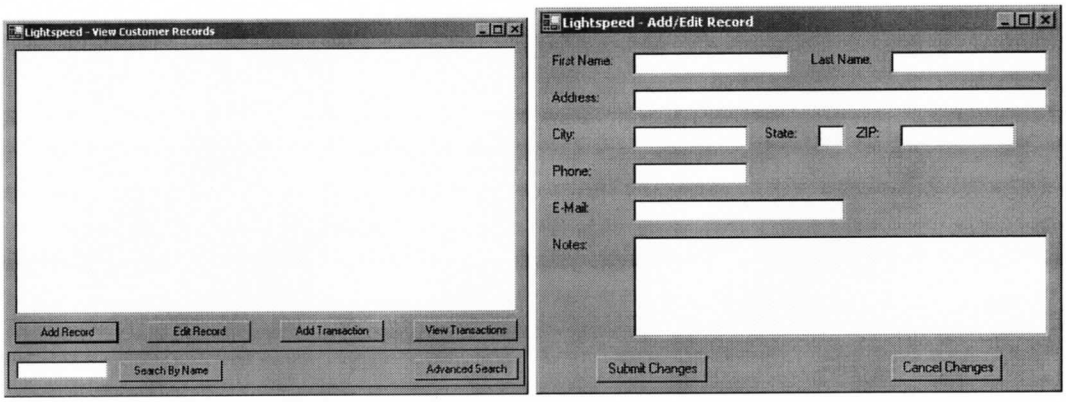
Scope of Work, Plan of Action, Activities

The budget for this project is extremely limited, and because of that it will be designed to operate on the resources that are currently available. This is the single most limiting factor of this project, and has resulted in many compromises in the planning stage. The most important compromise at this point is the usage of Microsoft Access as the DBMS. The client expressed some interest in MySQL, but in the end preferred to go with software which is currently installed and wouldn't require the only available workstation to run additional background services. Most other DBMS options are well out of the viable price range for the client.

The choice of Access as the DBMS is a large part of the reason why the ASP web interface will not be active upon project completion. Access is simply not secure enough to entrust with sensitive information online. The other major factor is the current lack of a server or web host. An HTTP server could easily be run on the workstation after some firewall configuration, but the client does not wish to risk leaving the only available computer open to attacks. Outside web hosts could provide the desired functionality, but

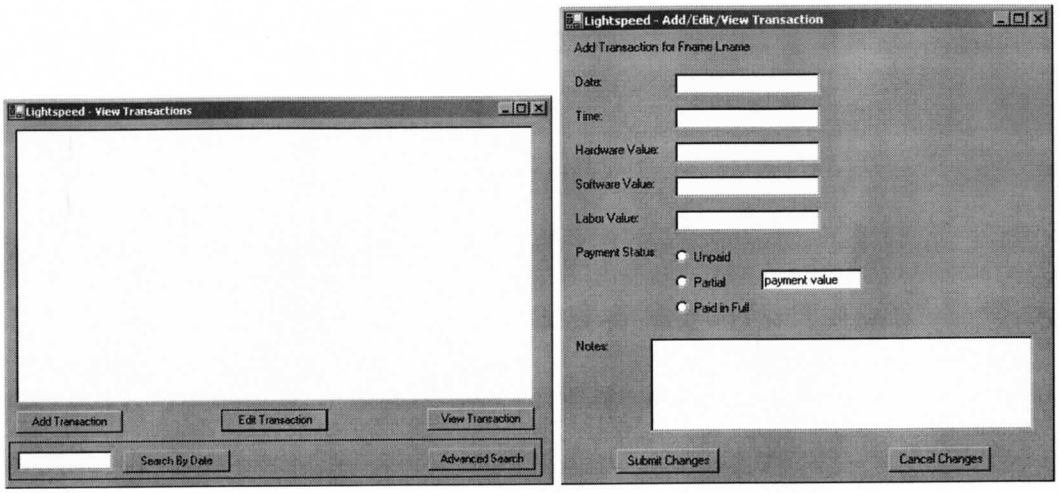
the client does not wish to incur another expense just to support the web interface at this time.

Much of the early Graphical User Interface (GUI) design is complete. Consultation with the client has resulted in an 'alpha' of the GUI for all screens except for the Advanced Search, which has been the subject of much debate and remains too uncertain to effectively illustrate. All of the alpha GUI screenshots shown below could be changed at any point, but the overall layout is expected to remain the same. Development of the Visual Basic.NET interface will be an iterative process involving prototype releases followed by testing of those prototypes.



The first iteration will be the customer information interface. This interface will consist of two forms and will allow for updating, adding, and viewing of customer information stored in the database. In addition, it will allow for simple querying based on customer name. This iteration is going to be created first because the client wishes to have a usable customer information database as soon as possible. Also, it is a logical choice for the first iteration since the transactional information is dependent upon the

customer information. Part of the testing of this iteration will be the beginnings of data entry into the database as well as the verification that this data entry has been successful.



The second iteration will be the transaction information interface. Like the customer information interface, this will consist of two forms and allow for updating, inserting, and viewing of transactional information stored in the database. It will also allow for simple searching by date. Testing of this iteration will mostly consist of inserting and updating existing transactions from paper records, while ensuring that these records remain properly recorded in the database.

The final iteration will be the advanced searching features. This will consist of one or two forms, providing many querying options to the user to maximize the functionality of the database system. The main goal of this iteration is to enable advanced queries which will allow the user to easily judge the costs associated with an ongoing service contract, perhaps to be used in later contract negotiations. Otherwise, the intent is to provide enough functionality that the user can always find the desired record or records, assuming that the user can provide enough information to do so effectively.

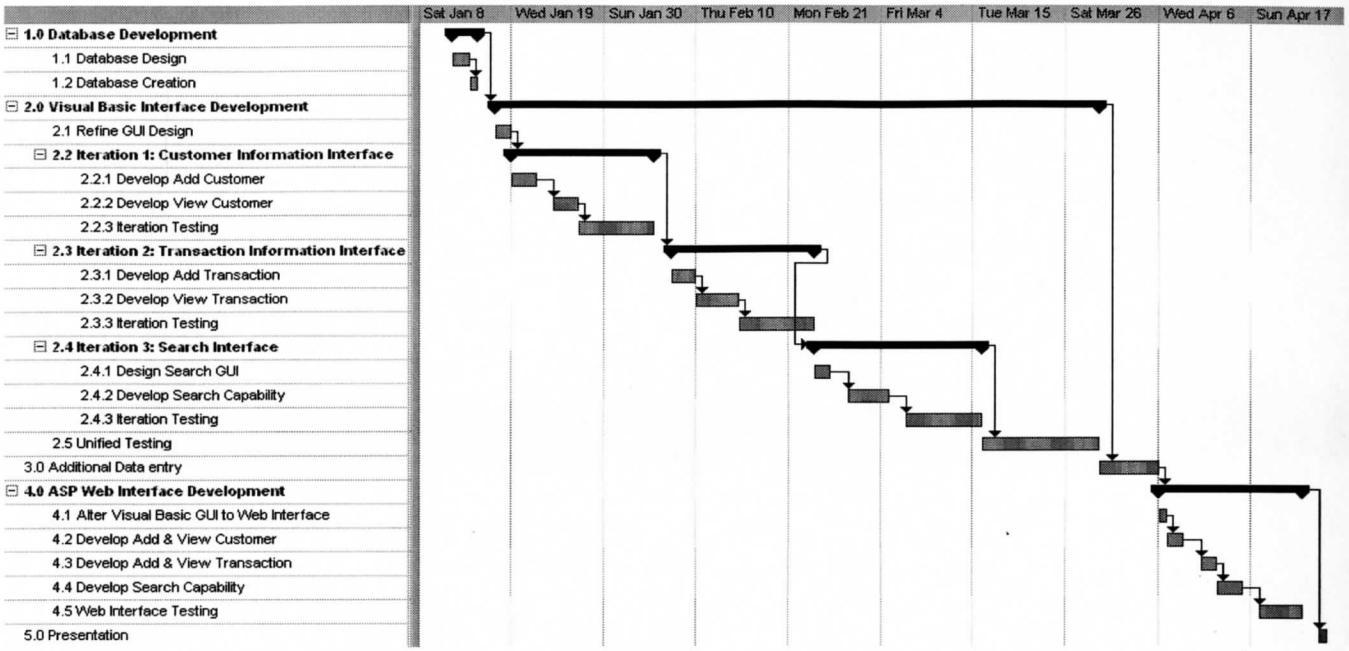
The testing of this iteration will consist of running a large variety of queries to ensure that the desired functionality has been achieved.

The development of the ASP.NET web interface will take place after the Visual Basic.NET interface and will largely draw upon that earlier work. As such, only one release is planned for the web interface, and it will be as similar as possible to the Visual Basic interface in order to avoid potential confusion and to lessen the learning curve associated with the web interface. The web interface will be created and tested on a separate copy of the database on a private server away from the client's site. This interface will not be publicly available and will not be connected to the active database in any way to prevent any possible security concerns.

Work Breakdown Structure and Gantt Chart

1.0 Database Development
    1.1 Database Design
    1.2 Database Creation
2.0 Visual Basic Interface Development
    2.1 Refine GUI Design
    2.2 Iteration 1: Customer Information Interface
        2.2.1 Develop Add Customer
        2.2.2 Develop View Customer
        2.2.3 Iteration Testing
    2.3 Iteration 2: Transaction Interface
        2.3.1 Develop Add Transaction
        2.3.2 Develop View Transaction
        2.3.3 Iteration Testing
    2.4 Iteration 3: Search Interface
        2.4.1 Design Search GUI
        2.4.2 Develop Search Capability
        2.4.3 Iteration Testing
    2.5 Unified Testing
3.0 Additional Data Entry
4.0 ASP Web Interface Development
    4.1 Alter Visual Basic GUI to Web Interface
    4.2 Develop Add & View Customer
    4.3 Develop Add & View Transaction

4.4 Develop Search Capability
4.5 Web Interface Testing
5.0 Presentation



Deliverables

The project deliverables include:

- A sample database based upon the final design and including valid data for testing purposes.
- The Visual Basic.NET interface code and executable, to be used with the sample database.
- The ASP.NET script used for the web interface. Since the development server will not be available to the public, an active webpage will not be deliverable.

These deliverables may be supplemented by additional project documentation, including program documentation and design documents created during application or database design.